การจำลองการใช้โทรศัพท์มือถือในการเข้าสู่และออกจากระบบรถไฟฟ้า

SIMULATION OF ACCESSING AND EXITING ELECTRIC TRAIN SYSTEM VIA MOBILE PHONE

โดย

นายวินน์ วรวุฒิคุณชัย

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2548

การจำลองการใช้โทรศัพท์มือถือในการเข้าสู่และออกจากระบบรถไฟฟ้า

SIMULATION OF ACCESSING AND EXITING ELECTRIC TRAIN SYSTEM VIA MOBILE PHONE

โดย

นายวินน์ วรวุฒิคุณชัย 44010452

อาจารย์ที่ปรึกษา

รศ.ดร. กอบชัย เดชหาญ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2548

ปริญญานิพนธ์ปีการศึกษา 2548

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การจำลองการใช้โทรศัพท์มือถือในการเข้าสู่และออกจากระบบรถไฟฟ้า

SIMULATION OF ACCESSING AND EXITING ELECTRIC TRAIN SYSTEM VIA MOBILE PHQNE

ผู้จัดทำ

นายวินน์ วรวุฒิคุณชัย 44010452

..... อาจารย์ที่ปรึกษา

(รศ.ดร. กอบชัย เดชหาญ)

การจำลองการใช้โทรศัพท์มือถือในการเข้าสู่และออกจากระบบรถไฟฟ้า

SIMULATION OF ACCESSING AND EXITING ELECTRIC TRAIN SYSTEM VIA MOBILE PHONE

โดย นายวินน์ วรวุฒิคุณชัย 44010452

อาจารย์ที่ปรึกษา รศ.ดร. กอบชัย เดชหาญ

บทคัดย่อ

เนื่องด้วยในปัจจุบันการเดินทางด้วยรถไฟฟ้าได้รับความนิยมเป็นอย่างมาก การซื้อบัตรโดยสารจากตู้ซื้อบัตรอาจจะไม่สะดวกและรวดเร็วเพียงพอในการรองรับผู้โดยสารโดยเฉพาะอย่างยิ่งในชั่วโมงเร่งด่วน โครงงานนี้ได้นำเสนอการจำลองการใช้โทรศัพท์มือถือในการเข้าสู่และออกจากระบบรถไฟฟ้าแทนการใช้บัตรโดยสาร โดยได้นำเอาเทคโนโลยีใหม่ๆอาทิ บลูทูธ มาประยุกต์ใช้ในการออกแบบและสร้างโครงงาน ทั้งนี้เพื่อเพิ่มความสะดวกสบายแก่ผู้ใช้บริการ

ABSTRACT

As the popularity in electric train transportation, especially in rush hours, purchasing tickets from ticket machines seems to be inconvenient and not fast enough to serve numerous passengers. For the more convenience in traveling, this simulation of accessing and exiting electric train system via mobile phone using modern technologies such as Bluetooth technology project has been established.

กิตติกรรมประกาศ

ปริญญานิพนธ์นี้จะสำเร็จลุล่วงไม่ได้ถ้าขาดความกรุณาในการให้คำปรึกษา แนะนำ ตลอดจนสนับสนุนของ รศ.ดร.กอบชัย เดชหาญ อาจารย์ที่ปรึกษา ขอกราบขอบพระคุณท่านอาจารย์อย่างสูง ขอกราบขอบคุณอาจารย์ทุกท่านที่ได้ประสิทธ์ประสาทวิชาความรู้ รวมถึงขอขอบคุณธุรการภาควิชา พี่ๆ น้องๆ และเพื่อนๆทุกคนที่ได้คอยช่วยเหลือ และให้กำลังใจข้าพเจ้ามาตลอด

สุดท้ายนี้ความดีอันบังเกิดจากปริญญานิพนธ์ฉบับนี้ขอมอบให้แก่ บิดา มารดา และผู้มีพระคุณทุกคนของข้าพเจ้า

สารบัญ

สารบัญรูปภาพ

บทที่ 1
บทนำ

1.1 ความเป็นมาของโครงงาน

เนื่องด้วยในปัจจุบัน มีผู้ที่นิยมเดินทางโดยสารด้วยรถไฟฟ้าเป็นจำนวนมาก อีกทั้งแนวโน้มการใช้บริการรถไฟฟ้านั้นได้เพิ่มขึ้นเรื่อยๆอย่างต่อเนื่อง ระบบการซื้อบัตรโดยสารรถไฟฟ้าในขณะนี้ ผู้โดยสารจะต้องซื้อบัตรโดยสารจากตู้หยอดเหรียญ ในช่วงชั่วโมงเร่งด่วนตู้หยอดเหรียญเพื่อซื้อตั๋วมีไม่เพียงพอที่จะบริการผู้โดยสารได้อย่างรวดเร็ว ผู้โดยสารจะต้องต่อแถวรอเพื่อซื้อบัตรโดยสารอันส่งผลให้เสียเวลาในการซื้อ และนำมาสู่การเดินทางล่าช้าออกไป ถึงแม้ว่าทางรถไฟฟ้าจะมีบัตรโดยสารรายเดือนอีกประเภทไว้คอยให้บริการแก่ผู้ที่โดยสารรถไฟฟ้าเป็นประจำก็ตาม แต่นั่นก็ทำให้เป็นภาระแก่ผู้โดยสารในการที่จะต้องพกพาบัตรเพิ่มขึ้นอีกใบ อีกทั้งทางผู้ให้บริการรถไฟฟ้าก็จะต้องมีค่าใช้จ่ายในการผลิตบัตรโดยสารด้วย ด้วยเหตุนี้ประกอบกับเทคโนโลยีของโทรศัพท์มือถือในปัจจุบัน ผู้จัดทำจึงได้คิดที่จะเพิ่มความสามารถให้กับโทรศัพท์มือถือให้สามารถใช้แทนบัตรโดยสารแทนในการเข้าออกระบบรถไฟฟ้าอันจะอำนวยความสะดวกแก่ผู้ใช้บริการรถไฟฟ้า

1.2 วัตถุประสงค์

1. เพื่ออำนวยความสะดวกสบายแก่ผู้โดยสารรถไฟฟ้า
2. เพื่อลดค่าใช้จ่ายในการผลิตบัตรโดยสาร
3. เพื่อเพิ่มความสามารถให้กับโทรศัพท์มือถือ

1.3 ขอบเขตของโครงงาน

1.ออกแบบระบบจำลองการใช้โทรศัพท์มือถือเปิดประตูทางเข้ารถไฟฟ้า โดยสามารถให้โทรศัพท์มือถือเปิดประตูจำลองที่อยู่บนคอมพิวเตอร์ได้

2. หักยอดเงินจากฐานข้อมูลที่สร้างไว้ตามระยะทางที่กำหนด

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1.สามารถที่จะพัฒนาโครงงานนี้ขึ้นเรื่อยๆจนสามารถที่จะนำไปใช้งานได้จริงในการอำนวยความสะดวกแก่ผู้ที่โดยสารรถไฟฟ้า ในอนาคต

2.สามารถนำเอาความรู้ที่ได้รับจากการทำโครงงานนี้ ไปพัฒนาประยุกต์สร้างโครงงานอื่นๆได้อีกหลายอย่างเป็นต้นว่า ทำให้โทรศัพท์มือถือเป็นรีโมทคอนโทรล โดยใช้บลูทูธในการส่งสัญญาณ ระบบช่วยเหลือคนพิการทางสายตา

3.นำความรู้จากการศึกษาการเขียนโปรแกรมและการออกแบบฐานข้อมูล ไปใช้ในการทำงานภายภาคหน้า

บทที่ 2
ทฤษฎีและหลักการพื้นฐาน

2.1 จาวาบลูทูธ (Java Bluetooth)

2.1.1 หลักการพื้นฐานจาวาบลูทูธ

เราสามารถสร้างโครงข่ายแอดฮอคเน็ตเวิร์ค (Adhoc network) ระหว่างอุปกรณ์บลูทูธ สำหรับหลักการพื้นฐานสำหรับจาวาบลูทูธนั้นจะมีดังต่อไปนี้

- การค้นหาอุปกรณ์ (Device inquiry)
- การค้นหาชื่อ (Name discovery)
-การค้นหาเซอร์วิซ (Service discovery)
-การทำงานของบลูทูธเสลฟ และมาสเตอร์ (Bluetooth master/slave roles)
-รหัสเฉพาะยูนิเวอร์เซิล (Universally Unique Identifiers)
-ช่องและแพคเก็ตแอลทูแคป (L2CAP channels and packets)
-อา เอฟ คอม (RFCOMM)

- การค้นหาอุปกรณ์

อุปกรณ์บลูทูธนั้นทำให้เราสามารถสร้างโครงข่ายแอดฮอค เพื่อการสร้างโครงข่ายนี้ อุปกรณ์บลูทูธจำเป็นที่จะต้องสามารถค้นหาอุปกรณ์บลูทูธตัวอื่น ซึ่งเรียกว่าการค้นหาอุปกรณ์ อุปกรณ์ที่ถูกค้นพบจะถูกระบุโดยแอดเดรสบลูทูธ (Bluetooth device address)

- การค้นหาชื่อ

นอกเหนือจากแอดเดรสบลูทูธแล้ว อุปกรณ์แต่ละตัวอาจยังมีชื่อเรียก (Friendly name) ซึ่งง่ายกว่าในการจดจำ การค้นหาชื่อเป็นกระบวนการในการจดจำชื่อของอุปกรณ์

- การค้นหาเซอร์วิซ

การค้นหาอุปกรณ์อย่างเดียวไม่เพียงพอที่จะใช้เซอร์วิซของอุปกรณ์บลูทูธตัวอื่น แอปพลิเคชั่นบนอุปกรณ์หนึ่งจะต้องค้นหาเซอร์วิซของอุปกรณ์บลูทูธอีกตัว ทันทีที่พบเซอร์วิซแล้วแอปลิเคชั่นก็จะสามารถติดต่อกันและใช้งานได้

- การทำงานของบลูทูธเสลฟ และมาสเตอร์

สำหรับการเชื่อมต่อระหว่างอุปกรณ์บลูทูธ 2 ตัว อุปกรณ์ตัวหนึ่งจะเป็นมาสเตอร์ส่วนอีกตัวหนึ่งจะเป็นเสลฟ ฟรีเควนซี่ฮอปปิ้งจะใช้เพื่อลดปัญหาการรบกวนกัน จะมีสัญญาณนาฬิกาและบลูทูธแอดเดรสเพื่อคำนวณลำดับความถี่ในการฮ๊อป ในกรณีที่เชื่อมต่อระหว่างแค่อุปกรณ์ 2 ตัว ไม่จำเป็นว่าตัวไหนจะเป็นมาสเตอร์หรือตัวไหนจะเป็นเสลฟ

- รหัสเฉพาะยูนิเวอร์เซิล

เป็นรหัส 128 บิต เราจะต้องกำหนดรหัสนี้เองสำหรับแอพลิเคชั่นที่เราสร้างขึ้น

- ช่องและแพคเก็ตแอลทูแคป

แอลทูแคปเป็นโปรโตคอลสื่อสารชั้นต่ำสุดที่ใช้โดยมิดเลทแอปพลิเคชั่น เราอาจจะใช้แอลทูแคปโปรโตคอลในแอพพลิเคชั่นถ้าเราต้องการควบคุมการส่งไบท์

- อา เอฟ คอม

โพรไฟล์บลูทูธพอร์ทอนุกรม จะใช้เพื่อสร้างการติดต่อระหว่างอุปกรณ์ 2 ตัว โปรโตคอล อา เอฟ คอม จะสร้างการสื่อสารอนุกรมและจัดเตรียมโฟล์คอนโทรล เราจะใช้ อา เอฟ คอม สำหรับ การติดต่อสตรีม

2.1.2 JSR-82

JSR-82 เป็น Official Java Bluetooth API ซึ่งถูกกำหนดมาตรฐานโดย JSR-82 Expert Group JSR-82 จะประกอบไปด้วย package จำนวน 2 package คือ

1. Javax.bluetooth (ประกอบไปด้วย 13 class สำหรับสร้างการสื่อสารไร้สายด้วย Bluetooth protocol)

2. Javax.obex (ประกอบไปด้วย 8 class สำหรับใช้ในการส่งผ่าน object ระหว่างอุปกรณ์)

2.2 จาวาเน็ตเวิร์คกิ่ง (Java Networking)

ใช้ในส่วนสร้างการติดต่อระหว่างสถานีกับสถานีกลาง

2.2.1 กล่าวนำ

Computer network คือกลุ่มเครือข่ายของคอมพิวเตอร์หลายเครื่องที่เชื่อมต่อกันด้วยตัวกลางบางอย่าง ทำให้สามารถรับส่งข้อมูลระหว่างกันได้ คอมพิวเตอร์ที่อยู่ในเครือข่ายเดียวกัน อาจมีโครงสร้างและระบบปฏิบัติการแตกต่างกันได้ แต่สามารถติดต่อกันได้เนื่องจากใช้โปรโตคอล (protocol)ในการติดต่อสื่อสารเดียวกัน โดยที่โปรโตคอล คือกฎเกณฑ์ที่กำหนดขั้นตอนในการติดต่อ แลกเปลี่ยนข้อมูลระหว่างกัน รวมทั้งรูปแบบของข้อมูลที่ถูกส่งไปมา โดยปรกติเรามักใช้โปรโตคอลที่เป็นที่นิยมอยู่แล้ว ซึ่งมีกฎเกณฑ์เป็นมาตรฐานคนทั่วไปรู้จักอย่างเช่น TCP, UDP และ HTTP เป็นต้น เพื่อช่วยให้การพัฒนาโปรแกรมเป็นอิสระจากภาษา ระบบปฏิบัติการ และอุปกรณ์สื่อสารที่ใช้

ในการนำส่งข้อมูลจากเครื่องหนึ่งไปสู่อีกเครื่องหนึ่ง ระบบเครือข่ายจะส่งข้อมูลออกไปเป็นส่วนย่อยๆ ทีละส่วนซึ่งเรียกว่าแพคเก็ต (packet) ด้านเครื่องที่เป็นผู้ส่งจะส่งแพคเก็ตผ่านชั้น (layer) ของโปรแกรมหลายชั้น เพื่อทำการเพิ่มเติมและปรับเปลี่ยนข้อมูลให้แพคเก็ตนั้นสามารถเดินทางผ่านทางตัวกลางไปสู่ผู้รับได้ ซึ่งอาจอยู่ในรูปสัญญาณไฟฟ้า แสง หรือคลื่นแม่เหล็กไฟฟ้า ขึ้นกับตัวกลาง เมื่อไปถึงเครื่องที่เป็นผู้รับแล้วฮาร์ดแวร์ที่เป็นโปรโตคอลชั้นล่างสุด ได้ข้อมูลจากตัวกลางมาแล้ว ก็จะส่งข้อมูลนั้นผ่านชั้นของโปรแกรมหลายชั้น เพื่อทำการเปลี่ยนคืนกลับเป็นแพคเก็ตดังเดิม

มีผู้ออกแบบชั้นของโปรโตคอลไว้หลายแบบ เช่น โอเพนซิสเต็มอินเตอร์คอนเน็คชั่น (Open System Interconnect) (OSI) แบ่งชั้นโปรโตคอล ออกเป็น 7 ชั้น คือ แอพพลิเคชั่น (application), พรีเซนเทชั่น (presentation), เซชชั่น (session), ทรานซพอร์ท (transport), เน็ตเวิร์ค (network), ดาต้าลิงค์ (data-

link) และ ฟิสิคัล (physical) ส่วน อินเตอร์เน็ตโปรโตคอล (Internet Protocol) (IP) แบ่งชั้นโปรโตคอล ไว้เป็น 5 ชั้น คือ แอปพลิเคชั่น ทรานซพอร์ท เน็ตเวิร์ค ดาต้าลิงค์ และฟิสิคัล

ในส่วนนี้จะอธิบายการสร้างโปรแกรม client และ server เพื่อส่งข้อมูลระหว่างเครื่องคอมพิวเตอร์ที่อยู่ในเครือข่ายด้วยภาษาจาวา ซึ่งนอกจากจะไม่ขึ้นกับแพลตฟอร์ม แล้วผู้เขียนโปรแกรมยังไม่จำเป็นต้องเข้าใจรายละเอียดเกี่ยวกับชั้นโปรโตคอล ฮาร์ดแวร์ หรือตัวกลางที่ใช้ในการสื่อสาร เนื่องจากกลไกที่จำเป็นในการส่งข้อมูลผ่านระบบเครือข่ายถูกซ่อนรายละเอียด (encapsulated) อยู่ภายในคลาสของ java.net package ซึ่งจะอธิบายดังต่อไปนี้

2.2.2 ไอเนตแอดเดรส (InetAddress)

ในการส่งข้อมูลระหว่างเครื่องคอมพิวเตอร์ เราต้องสามารถระบุ (identify) เครื่องที่จะเป็นผู้รับและผู้ส่ง เราเรียกว่าคอมพิวเตอร์แต่ละเครื่องที่ต่ออยู่ในระบบอินเตอร์เน็ตว่าโฮส(host)โดยปกติทุกโฮสจะถูกกำหนดเลขที่ประจำตัวเรียกว่าอินเตอร์เน็ตแอดเดรส (Internet address) หรือไอพีแอดเดรส (IP address) มีขนาด 4 ไบท์ เพื่อความสะดวกเราจะเขียนไอพีแอดเดรสเป็นเลขจำนวนเต็มบวก 4 ตัว แบ่งแยกกันด้วยจุด '.' โดยเลขจำนวนเต็มตัวหนึ่งสำหรับค่าหนึ่งไบต์จึงมีค่าได้ตั้งแต่ 0 ถึง 255 ตัวอย่างเช่น 203.146.19.3 โดยปกติบริษัทผู้ให้บริการเข้าสู่อินเทอร์เน็ต (Internet Service Provider) จะเป็นผู้กำหนดไอพีแอดเดรสให้แก่เครื่องของเรา หรือถ้าเครื่องของเราอยู่ในระบบเครือข่ายของมหาวิทยาลัย บริษัทหรือองค์กรขนาดใหญ่ ก็จะมีผู้ดูแลระบบ (system administrator) เป็นบุคคลที่มีหน้าที่ดูแลรักษาระบบ ทำการจัดสรรไอพีแอดเดรสให้แก่เครื่องแต่ละเครื่อง หากเครื่องของเราไม่ได้ต่อกับระบบเครือข่าย และยังไม่ได้กำหนดไอพีแอดเดรสไว้ เราก็ยังสามารถอ้างถึงเครื่องของเราเองได้โดยใช้ลูปแบ้ไอพีแอดเดรส (loopback IP address) คือ 127.0.0.1 ซึ่งเป็นเบอร์ที่จะถูกตีความเป็นเครื่องที่กำลังทำงานโดยดีพอร์ท

ไอพีแอดเดรสเป็นตัวเลขจึงเหมาะสำหรับถูกจัดการโดยเครื่องคอมพิวเตอร์ แต่ผู้ใช้ที่เป็นมนุษย์จะมีปัญหาในการจดจำตัวเลข ดังนั้นเพื่อช่วยให้เราสามารถระบุเครื่องคอมพิวเตอร์ได้ด้วยชื่นที่จำง่ายกว่า จึ้มีการกำหนดโดเมนเนมไว้คู่กับไอพีแอดเดรสเช่นไอพีแอดเดรส 203.149.8.2 มีโดเมนเนมเป็น cs.tu.ac.th ทำให้เราสามารถอ้างถึงเครื่องนี้โดยใช้ 203.149.8.2 หรือ cs.tu.ac.th ก็ได้ แต่เราจะใช้โดเมนเนมก็ต้องมีดีเอ็นเอส (DNS) (Domain Name System) ซึ่งเป็นเซอฟเวอร์ที่ให้บริการ

ใน java.net package มีคลาส InetAddress สำหรับเก็บแสดงไอพีแอดเดรสซึ่งใช้ได้กับทั้งโปรโตคอล ทีซีพี (TCP) และ ยูดีพี (UDP) โดยปกติในอินแสตนท์(instance)ของคลาส InetAddress จะมีข้อมูลเกี่ยวกับไอพีแอดเดรสและอาจจะมีโดเมนเนมของไอพีแอดเดรสนั้นด้วยก็ได้ ขึ้นกับว่าอินแสตนท์นั้นถูกสร้างขึ้นโดยมีโดเมนเนมกำหนดให้หรือไม่ การที่โปรแกรมภาษาจาวา ใช้ไอเนตแอดเดรสแทนไอพีแอดเดรสในการอ้างอิงเครื่องๆหนึ่ง ก็เพื่อให้โปรแกรมไม่ขึ้นกับแพลตฟอร์ม ซึ่งอาจจะมีวิธีในการเก็บแสดงไอพีแอดเดรสที่แตกต่างกัน และในอนาคตคงต้องมีการเปลี่ยนแปลงไอพีแอดเดรสให้สามารถรองรับจำนวนเครื่องคอมพิวเตอร์ที่เพิ่มมากขึ้น เมื่อถึงเวลานั้นคลาส InetAddress จะถูกเปลี่ยนแปลงให้สนับสนุนมาตรฐานใหม่นั้น โดยที่โปรแกรมของเราไม่ต้องถูกเปลี่ยนแปลง อย่างไรก็ตาม ในตอนนี้เริ่มมี

การทดลองใช้มาตรฐาน IPv6 จึงมีการเพิ่มคลาส InetAddress สำหรับ IPv4 และคลาส Inet6Address สำหรับ IPv6

คลาส InetAddress ไม่มีพับบลิคคอนสตรัคเตอร์ (public constructor) แต่มีพับบลิคสเตติคคอนสตัคเตอร์เมธอด (public static factory method) สำหรับสร้างอินแสตนซ์ของคลาส อย่างเช่น

```
public static InetAddress getLocalHost( ) throws
UnknownHostException;
```

ซึ่งจะให้ ไอพีแอดเดรสของเครื่องที่ใช้ทำงาน หลังจากนั้น เราอาจใช้

```
public String getHostName( );
public String getHostAddress( );
```

ตัวอย่าง แสดงการใช้ getLocalHost( )

```
// GetLocalHostTest.java
import java.net.*;
class GetLocalHostTest {
      public static void main(String args[]){
            try {
                  InetAddress a = InetAddress.getLocalHost();
                  System.out.println("name: " + a.getHostName());
                  System.out.println("IP: " + a.getHostAddress());
            } catch (UnknownHostException e) {
                  System.out.println(e);
            }
      }
}
```

โปรแกรมนี้เรียกใช้ getLocalHost( ) ซึ่งได้ผลลัพธ์ออกมาเป็น InetAddress a จากนั้นนำมาเรียก getHostName( ) และ getHostAddress( ) แล้วพิมพ์ผลที่ได้ออกมา

2.2.3 ซอคเก็ต (Socket)

ในการส่งข้อมูลจากเครื่องคอมพิวเตอร์หนึ่งไปสู่อีกเครื่องหนึ่งในระบบเครือข่าย โปรแกรมด้านผู้ส่ง (sender) จะต้องนำข้อมูลที่ถูกส่งไปนั้น มาตัดออกเป็นส่วนย่อยๆ แล้วบรรจุลงในแพคเก็ตแต่ละแพคเก็ตจะมีส่วนประกอบสองส่วน ส่วนแรกคือเฮดเดอร์เป็นข้อมูลเกี่ยวกับแอดเดรสและพอร์ทของผู้รับและผู้ส่ง รวมทั้งข้อมูลเท่าที่จำเป็นในการนำแพคเก็ตมาประกอบกันเป็นข้อมูลดังเดิม อีกส่วนเรียกว่าเพโหลด (payload) คือข้อมูลย่อยที่จะถูกส่งไปนั่นเอง ผู้เขียนโปรแกรมต้องสร้างลงไปสู่ชั้นตัวกลางเพื่อเดินทางไปในระบบเครือข่าย ที่ด้านผู้รับ ต้องทราบวิธีรับแพคเก็ตจากระบบเครือข่าย ขึ้นมาประกอบเป็นลำดับที่ถูกต้อง แล้วจึงดึงข้อมูลออกมา จะเห็นว่าการเขียนโปรแกรมรับส่งข้อมูลผ่านระบบเครือข่ายเป็นเรื่องยุ่งยากมากและต้องใช้ผู้เชี่ยวชาญในการสร้างโปรแกรมแบบนี้ จนกระทั่ง Berkeley UNIX ได้มีการเสนอซอคเก็ตขึ้น (socket) เพื่อใช้ในการเขียนโปรแกรมส่งข้อมูลผ่านระบบเครือข่าย โดยที่ซอคเก็ตจะซ่อน

รายละเอียดเกี่ยวกับแพคเก็ต รวมทั้งวิธีการติดต่อกับระบบเครือข่ายไปจากผู้เขียนโปรแกรมแล้วให้ผู้เขียนโปรแกรม สามารถเขียนโปรแกรมส่งและรับข้อมูลผ่านระบบเครือข่ายได้เหมือนกับการเขียนและอ่านข้อมูลจากสตรีมเมื่อมีซอคเก็ตแล้ว การเขียนโปรแกรมส่งข้อมูลผ่านระบบเครือข่ายง่ายขึ้นอย่างมาก จึงแพร่หลายจาก UNIX ไปสู่ระบบปฏิบัติอื่นๆ เช่น Windows และ Macintosh รวมทั้งยังอาจเขียนโปรแกรมเกี่ยวกับซอคเก็ตด้วยภาษาต่างๆโดยเริ่มจากภาษา C ไปสู่ C++, VB และ Java ซึ่งเราจะกล่าวถึงการใช้ ซอคเก็ตในภาษาจาวาดังต่อไปนี้

ในแพคเก็ต java.net มีคลาสซอคเก็ตสำหรับสร้างซอคเก็ตซึ่งมีคลไกสำหรับ

- สร้างการติดต่อไปสู่เครื่องเป้าหมายที่ พอร์ทหนึ่ง โดยเครื่องเป้าหมายอาจเป็นผู้รับหรือส่งก็ได้
- สร้างสตรีนสำหรับอ่านหรือเขียนข้อมูลไปที่เครื่องเป้าหมาย
- ปิดการติดต่อไปสู่เครื่องเป้าหมาย

คลาส Socket มีคอนสตรัคเตอร์หลายตัว เช่น

```
public Socket (String,int) throws UnknownHostException, IOException;
public Socket (InetAddress, int) throws IOException;
```

โดยจะสร้างซอคเก็ตไปยังเครื่องโฮสที่ระบุพารามิเตอร์ตัวแรกที่เป็น String หรือ InetAddress โดยติดต่อเข้าไปที่พอร์ทเลขที่ระบุด้วยพารามิเตอร์ตัวที่สองที่เป็น int

ในการสร้างอินแสตนท์ของคลาส Socket โดยการ new คอนสตรัคเตอร์เราต้องระบุค่าโฮสซึ่งเป็นเครื่องเป้าหมายที่ซอคเก็ตจะติดต่อไป และค่าพอร์ทที่เครื่องเป้าหมายนั้นเปิดรอรับไว้ หากที่เครื่องเป้าหมายไม่ได้เปิดพอร์ทเบอร์นั้นไว้จะเกิด IOExpeption โดยทั่วไปเราอาจจะระบุค่าโฮสเป็น string ซึ่งจะใช้รูปแบบที่เป็น IP address คือ "xxx.xxx.xxx.xxx" หรือเป็นโดเมนเนมก็ได้ หากค่าของโฮสนั้นผิดพลาดหรือไม่มีอยู่จริงก็จะเกิด UnknownHostException สังเกตว่า ถ้าเราระบุโฮสด้วย InetAddress เป็นพารามิเตอร์ตัวแรก โปรแกรมจะทำงานได้เร็วกว่าระบุ โฮสด้วย string เล็กน้อย

เมื่อสร้างซอคเก็ตจะมีการติดต่อไปที่เครื่องเป้าหมายทันที โดยใช้ ทีซีพี/ไอพี หลังจากเกิดการติดต่อแล้ว เครื่องเริ่มต้นจะสามารถขอ input stream และ output stream จาก socket เพื่อทำการอ่านและเขียนไปที่เครื่องเป้าหมายดังนั้นต้องมีข้อตกลงระหว่างเครื่องทั้งสองว่า จะมีกติกาในการรับส่งข้อมูลอย่างไร เช่น ใครจะเขียนหรืออ่านก่อน เป็นต้น และข้อมูลต้องมีรูปแบบใด เราเรียกข้อตกลงนี้ว่า โปรโตคอล เรากำหนดให้แต่ละ พอร์ทต้องมีโปรโตคอลที่แน่นอนเพื่อให้เราสามารถติดต่อเข้าไปที่พอร์ทนั้นได้ ถ้าเข้าใจโปรโตคอลแต่ละโปรแกรมนั้นอาจถูกเขียนด้วยภาษาใดหรือทำงานบนแพลตฟอร์มใดก็ได้ ค่าของพอร์ทต้องเป็นเลขจำนวนเต็มที่มีค่าตั้งแต่ 1 ถึง 65,535 แต่ในเครื่องทั่วไป พอร์ทเบอร์ตั้งแต่ 1 ถึง 1024 จะถูกกำหนดเป็นพอร์ทที่มีโปรโตคอลมาตรฐานไว้แล้ว เช่น เบอร์ 21 เป็น ftp, 23 เป็น telnet, 25 เป็น smtp, 80 เป็น http เป็นต้น และยังมีโปรแกรมของผู้ผลิตบางเจ้านำพอร์ทที่มีค่าสูงกว่า 1024 ไปใช้แล้ว ดังนั้นหากเราจะเปิดพอร์ทเพื่อทำการติดต่อกันเอง ควรเลือกค่าที่สูงกว่า 1024 ขึ้นไป และตรวจสอบด้วยว่ามีโปรแกรมอื่นที่ใช้พอร์ทเบอร์นั้นอยู่ก่อนหรือไม่

เราจะทดสอบการทำงานของ HelloServer โดยสร้างโปรแกรม Client ดังนี้

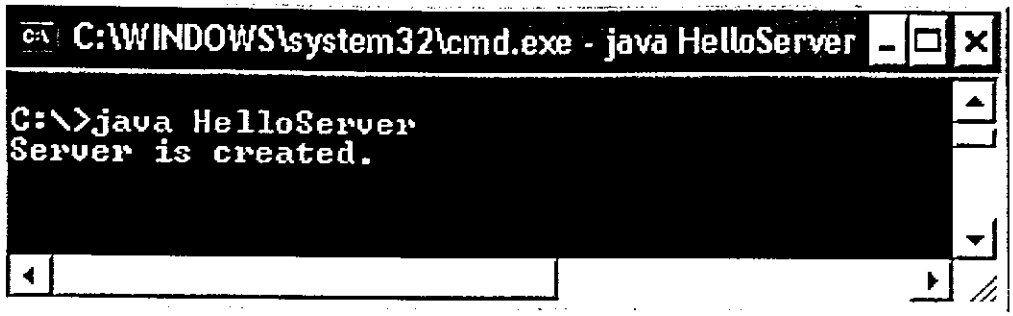
```
// Client.java
import java.io.*;
import java.net.*;
class Client
{
      public static void main(String args[]) throws Exception
      {
            Socket s = new Socket("localhost", 12345);
            BufferedReader br = new BufferedReader(new
            InputStreamReader (s.getInputStream()));
            System.out.println(br.readLine());
            br.close();
            s.close();
      }
}
```

โปรแกรมนี้จะสร้าง Socket s ไปที่เครื่องที่กำลังใช้งานคือ "localhost" และต่อไปที่ port 12345 ซึ่งเปิดไว้โดย HelloServer เราทราบว่าโปรโตคอลนี้จะส่งข้อความมาเป็นตัวอักษรบรรทัดเดียว จึงใช้ readLine ( ) จาก input stream ออกมาพิมพ์

โปรแกรม HelloServer

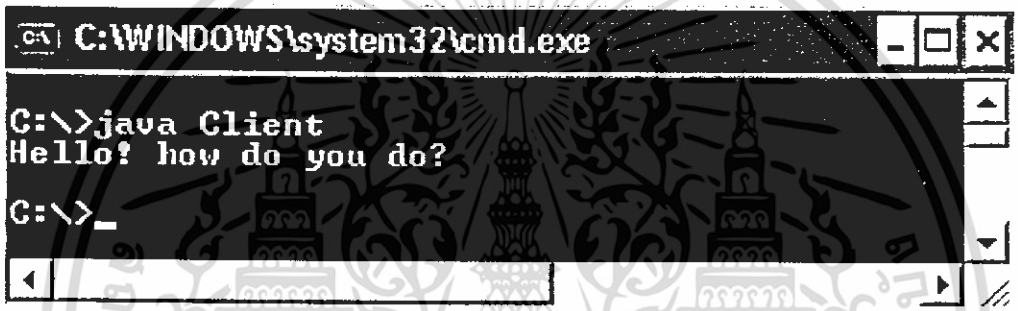
```
// HelloServer.java
import java.io.*;
import java.net.*;
class HelloServer
{
      public static void main(String args[]) throws Exception
      {
            ServerSocket ss = new ServerSocket(12345);
            System.out.println("Server is created.");
            Socket s = ss.accept();
            PrintStream op = new
            PrintStream(s.getOutputStream());
            op.println("Hello! how do you do?");
            s.close();
            ss.close();
      }
}
```

เราเริ่มทำงาน HelloServer ก่อน โดยเปิด command prompt และพิมพ์ java HelloServer ดังนี้



รูปที่ 2.1 แสดงการรันโปรแกรม HelloServer

จากนั้นเปิด command promt อีกทีแล้วพิมพ์ java Client จะได้ผลดังนี้



รูปที่ 2.2 แสดงการรันโปรแกรม Client

แล้ว HelloServer จะหยุดทำงานลง

2.3 Java Data Base Connection (JDBC)

ใช้ในส่วนเก็บฐานข้อมูลผู้ใช้บริการ รวมถึงรายละเอียดและอัตราค่าโดยสารของสถานีรถไฟฟ้า

2.3.1 เกริ่นนำ

ทุกธรกิจและองค์กรมีข้อมูลที่ต้องเก็บรักษา และนำออกมาใช้ ระบบฐานข้อมูลจึงเป็นสิ่งจำเป็นในโปรแกรมเชิงธุรกิจ แต่ผู้ผลิตระบบฐานข้อมูลมีมากมายหลายยี่ห้อ มีขนาดและความสามารถแตกต่างกัน วิธีเข้าใช้งานก็ต่างกันไปด้วย การเขียนโปรแกรมที่ต้องติดต่อเข้าใช้ฐานข้อมูลจึงยุ่งยากกว่าที่ควรจะเป็น ต่อมาผู้ผลิตระบบฐานข้อมูลส่วนใหญ่ยอมรับรูปแบบภาษาคิวรี่ (Structured Query Language) (SQL) เป็นภาษามาตรฐานสำหรับจัดการกับข้อมูล แต่อย่างไรก็ตาม ระบบฐานข้อมูลแต่ละยี่ห้อยังสนับสนุน SQL ที่ต่างกันอยู่บ้าง และก่อนที่โปรแกรมจะสามารถส่ง SQL ให้แก่ระบบฐานข้อมูลได้ ก็ต้องติดต่อกับระบบฐานข้อมูลให้ได้ก่อน ซึ่งวิธีติดต่อยังคงแตกต่างกันออกไป ในช่วงสิบปีที่ผ่านมา มีการพัฒนาในทางที่ดีเกิดขึ้นสองอย่าง ข้อหนึ่ง คือผู้ผลิตระบบฐานข้อมูลส่วนใหญ่ตกลงยอมรับ SQL-92 เป็นมาตรฐาน ทำให้โปรแกรมที่ใช้ SQL-92 สามารถส่งคำสั่งไปทำงานในระบบฐานข้อมูลไปเกือบทุกยี่ห้อ ข้อที่สอง คือ บริษัทผู้ผลิตซอฟแวร์หลายแห่งพยายามสร้างมาตรฐานสำหรับให้โปรแกรมติดต่อเข้าใช้งานระบบ

ฐานข้อมูล เพียงแค่บริษัทไมโครซอฟท์เจ้าเดียวก็มีหลายมาตรฐาน เช่น ODBC, ADOBD, DAO และ OLEDB เป็นต้น มาตรฐานเหล่านี้ช่วยให้โปรแกรมสามารถติดต่อเข้าใช้งานระบบฐานข้อมูลยี่ห้อต่างๆได้ด้วยวิธีเดียวกัน จึงไม่ต้องสร้างโปรแกรมสำหรับใช้งานระบบฐานข้อมูลแต่ละยี่ห้อ แต่ปัญหาคือมาตรฐานเหล่านี้ยังขึ้นกับภาษา และแพลทฟอร์มที่ใช้งานรวมทั้งบางมาตรฐานไม่ได้ถูกสร้างขึ้นมาเพื่อสนับสนุนการเรียกใช้จากภาษาเชิงวัตถุ

เมื่อตอนที่ภาษาจาวาออกมาใหม่ๆ ยังถูกถือว่าไม่ใช้ภาษาที่ใช้งานได้จริง เพราะยังไม่สามารถติดต่อกับระบบฐานข้อมูล ดังนั้นในปี 1996 บริษัทซันไมโครซิสเต็มจึงได้พัฒนาการติดต่อดาต้าเบสด้วยจาวา (Java Database Connection) (JDBC) โดยแบ่งออกเป็นสองส่วนคือ JDBC api เป็นชุดของคลาสและอินเทอร์เฟส เพื่อให้โปรแกรมภาษาจาวา ใช้ติดต่อกับระบบฐานข้อมูลใดๆ อีกส่วนหนึ่งคือ ข้อกำหนดของ JDBC ไดรเวอร์ เพื่อให้บริษัทผู้ผลิตระบบฐานข้อมูลนำไปสร้างไดรเวอร์ที่ติดต่อกับระบบฐานข้อมูลของเขา ทำให้โปรแกรมไม่ผูกติดกับระบบฐานข้อมูลที่ใช้ รวมทั้งไม่จำเป็นต้องทราบรายละเอียดวิธีติดต่อเข้าระบบฐานข้อมูลนั้น การติดต่อระหว่างโปรแกรมกับระบบฐานข้อมูลแสดงได้ดังในรูปนี้

รูปที่ 2.3 แสดงการติดต่อระหว่างโปรแกรมกับฐานข้อมูล

ผู้ผลิตระบบฐานข้อมูลแต่ละยี่ห้อจะสร้างไดรเวอร์ (Driver) สำหรับระบบฐานข้อมูลนั้น โดยไดรเวอร์ จะแปลคำสั่งจาก JDBC เป็นคำสั่งสำหรับเข้าใช้งานระบบฐานข้อมูล ส่วนโปรแกรมภาษา Java จะส่งคำสั่งสำหรับติดต่อกับระบบฐานข้อมูลเป็น SQL ไปที่ JDBC แล้ว JDBC เปลี่ยนคำสั่งนั้นเป็นคำสั่งสำหรับ drivers

JDBC นั้นมีกลไกพื้นฐาน 3 อย่างคือ

1. เริ่มต้นติดต่อกับระบบฐานข้อมูล
2. ส่งคำสั่ง SQL ไประบบฐานข้อมูล
3. รับผลลัพธ์จากระบบฐานข้อมูลเข้ามาจัดการ

2.3.2 การสร้างการเชื่อมต่อ

ระบบฐานข้อมูลโดยทั่วไปมีขั้นตอนการสร้างการติดต่อ รวมทั้งการส่งคำสั่ง SQL ออกไป และรับผลลัพธ์กลับมาเหมือนกัน เพียงมีชื่อไดรเวอร์และวิธีอ้างถึงชื่อดาต้าเบสที่แตกต่างกันไปในแต่ละยี่ห้อ

สมมติว่าจะมีตารางหนึ่งชื่อ STUDENT ถูกสร้างไว้แล้ว หรือสร้างโดยใช้โปรแกรมภาษาจาวา หรือคำสั่งของระบบฐานข้อมูลนั้นกำหนดว่ามีโครงสร้างดังนี้

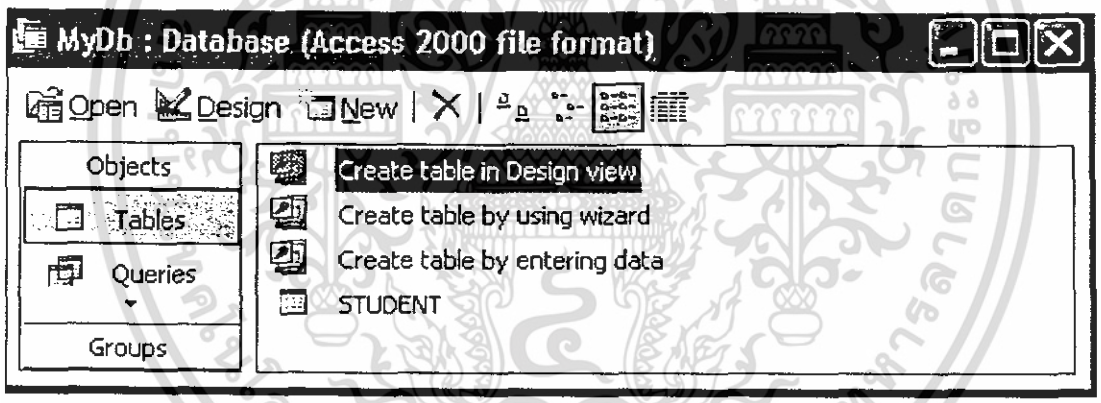
และให้มีข้อมูลเริ่มต้นจำนวนหนึ่ง สำหรับให้โปรแกรมทดสอบนำข้อมูลใน STUDENT table ออกมาพิมพ์ทาง standard output แต่ในตอนนี้เป็นการเตรียมการเพื่อเชื่อมต่อกับฐานข้อมูลเท่านั้น ดังนั้นเราจะแสดงโปรแกรมทดสอบเพื่อใช้งานไปก่อน แล้วจะอธิบายรายละเอียดของโปรแกรมในภายหลัง

การติดต่อกับ Microsoft Access

Microsoft ODBC (Open Database Connectivity) เป็น API ที่ถูกใช้อย่างแพร่หลาย สามารถติดต่อกับระบบฐานข้อมูลได้หลายยี่ห้อ และใช้ได้ในหลายแพลทฟอร์มแต่สาเหตุที่เราไม่เขียนโปรแกรมเพื่อเรียกใช้ ODBC ตรงๆ เองจากภาษาจาวาก็เพราะ ODBC ถูกเขียนขึ้นด้วยภาษา C มีปัญหาในเรื่องของความปลอดภัยของข้อมูลและความเข้ากันได้ของโปรแกรม ดังนั้นจึงมี JDBC-ODBC Bridge เพื่อเป็นสะพานเชื่อมต่อการทำงานจากโปรแกรมภาษาจาวาไปยัง ODBC Driver

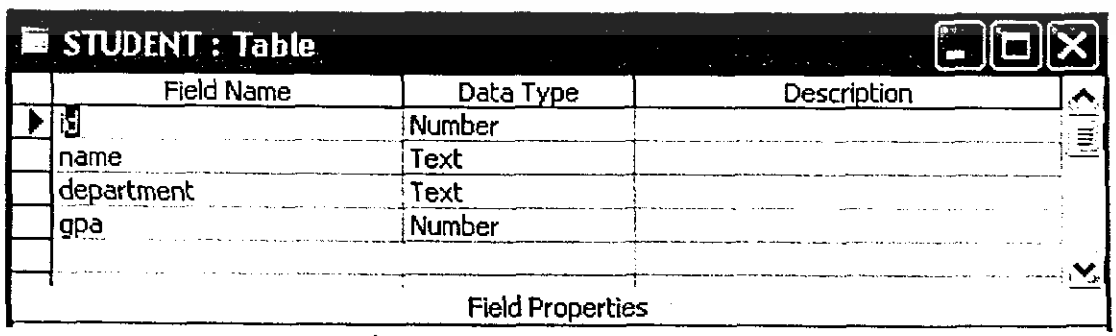
ขั้นตอนการติดตั้งไฟล์ดาต้าเบส เข้ากับ ODBC คือ

1. ใช้ Microsoft Access สร้าง database ชื่อ MyDb มี table ชื่อ STUDENT



รูปที่ 2.4 แสดงการใช้ Microsoft Access สร้างฐานข้อมูล

กดปุ่ม Design แล้วสร้าง STUDENT table ให้โครงสร้างดังนี้



รูปที่ 2.5 แสดงการสร้างตาราง STUDENT

และใส่ข้อมูล เพื่อใช้ในการทดสอบ ดังนี้

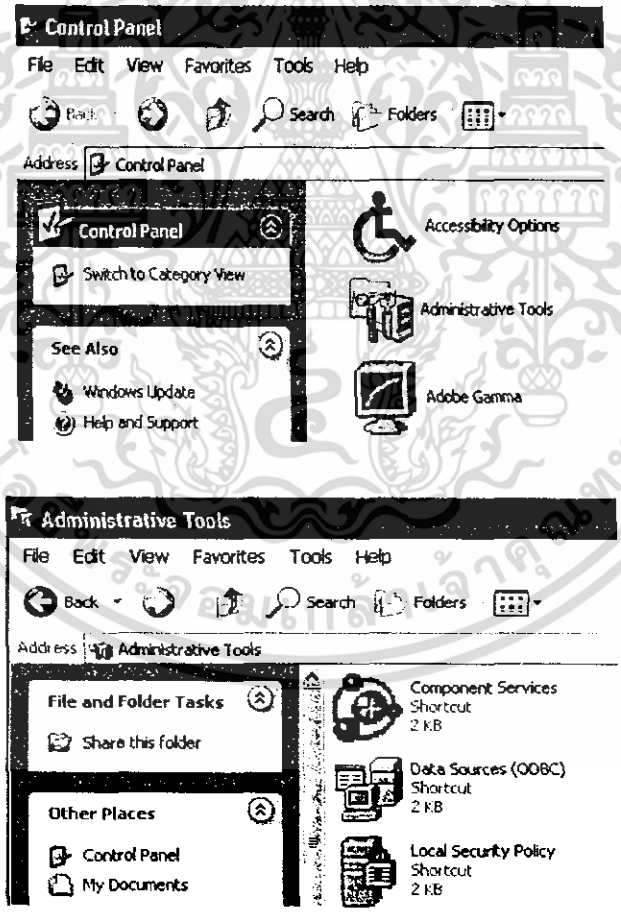
STUDENT : Table

id	name	department	gpa
7	Jame Bond	Comp Sci	2
123	John Rambo	Comp Sci	1
666	Jack Ripper	Biology	4
			0

รูปที่ 2.6 แสดงการใส่ข้อมูลลงในฐานข้อมูล

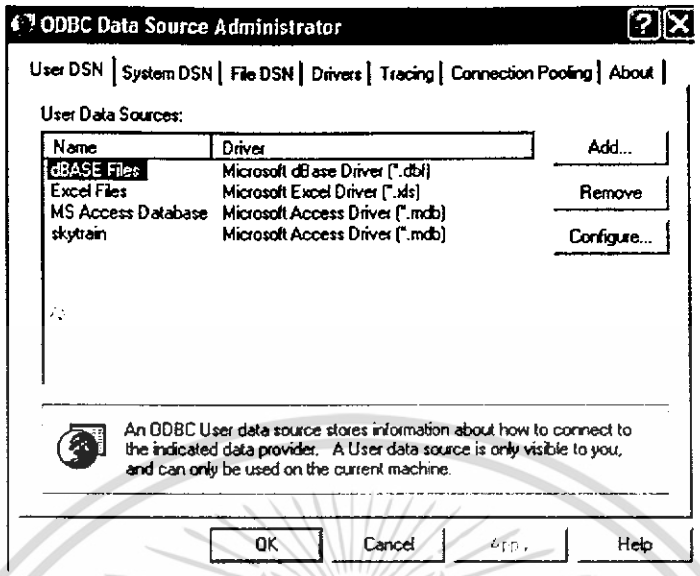
เซฟไฟล์นี้ไว้ที่ c:\Odbc\MyDb.mdb

2. เรียกโปรแกรม ODBC Data Source Administrator ให้ทำงาน (ใน Window XP) โดย เรียกจาก Control Panel-Administrative tools-DataSources(ODBC)



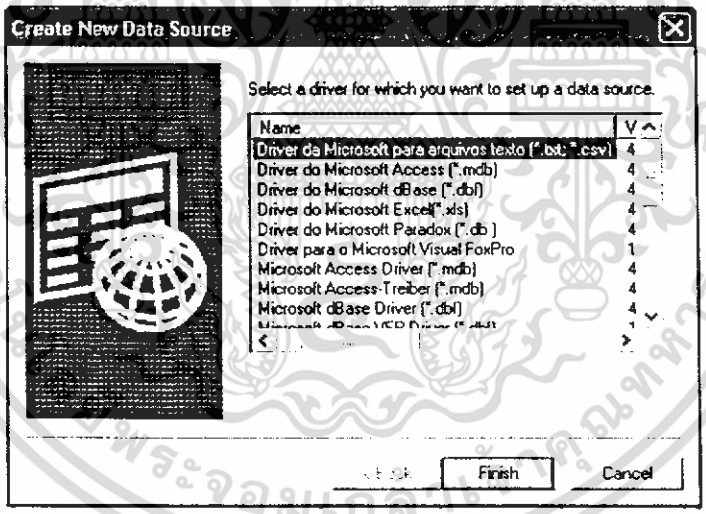
รูปที่ 2.7 แสดงการเรียกโปรแกรม ODBC Data Source Administrator

จะปรากฏ ODBC Data Source Administrator dialog ดังนี้



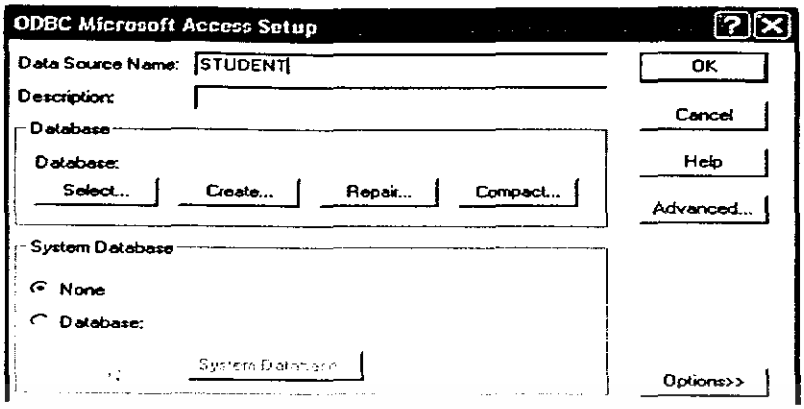
รูปที่ 2.8 แสดงหน้าต่าง ODBC Data Source Administrator

เลือก Tab User DSN แล้วเลือกปุ่ม Add จะปรากฏ Create New Data Source dialog



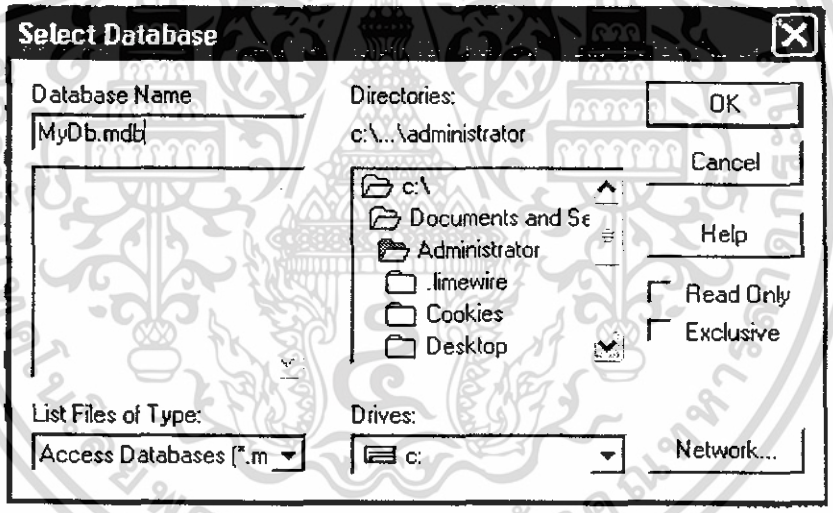
รูปที่ 2.9 แสดงหน้าต่างการสร้าง Data Source

เลือกชนิดดาต้าเบส ที่ ODBC ไดรเวอร์รู้จัก ในที่นี้ให้เลือก Microsoft Access Driver (*.mdb) แล้วกดปุ่ม Finish จะปรากฏ ODBC Microsoft Access Setup dialog



รูปที่ 2.10 ODBC Microsoft Access Setup dialog

ตั้งชื่อ Data Source Name เพื่อใช้อ้างถึงผ่านทาง JDBC-ODBC Bridge เป็นชื่อของ table ที่จะถูกเชื่อมต่อ ในกรณีนี้คือ STUDENT จากนั้นกดปุ่ม Select เพื่อเลือกชื่อไฟล์ database คือ MyDb.mdb ที่สร้างไว้ในข้อ 1. จะปรากฏ Select Database dialog ดังนี้



รูปที่ 2.11 Select Database dialog

เลือก Directories: ไปที่ c:\Odbc แล้วเลือก Database Name เป็น MyDb.mdb แล้วกดปุ่ม OK เพื่อจบการทำงานของทุกๆ dialogs ที่ค้างอยู่ เป็นการจบสิ้นการติดตั้ง จากนี้ไป ODBC จะรู้จัก data source name "STUDENT" ที่อยู่ในไฟล์ MyDb.mdb นี้จนกว่าจะถูก remove โดยใช้ ODBC Data Source Administrator dialog

โปรแกรมนี้ใช้สำหรับตรวจสอบว่าการสร้างและติดตั้ง ODBC ให้รู้จัก MyDb.mdb และ student table ถูกต้องหรือไม่

Statements

ระบบฐานข้อมูลเกือบทั้งหมดสนับสนุนการติดต่อเข้าใช้งานด้วย SQL แม้ว่าอาจจะมีความแตกต่างกันบ้าง JDBC กำหนด Statement เป็น interface สำหรับสร้างคลาสที่ทำหน้าที่ส่งประโยค SQL ไปสู่ระบบฐานข้อมูลผ่านทาง connection แล้วถ้าประโยค SQL นั้นมีผลลัพธ์ ก็จะสามารถรับผลลัพธ์นั้นกลับเข้าสู่โปรแกรมเป็น result set ซึ่งนำมาจัดการได้ในโปรแกรม เราสามารถใช้ Statement ได้สามแบบ คือ

- Statement สำหรับทำงานประโยค SQL ที่สร้างขึ้นเพื่อให้ครั้งเดียว
- PreparedStatement สำหรับทำงานประโยค SQL ที่มีรูปแบบเดิมหลายๆครั้ง
- CallableStatement สำหรับทำงาน stored procedures

โดยปกติเราไม่ต้องสร้างคลาสที่ Implements Statement เองแต่จะใช้

```
public abstract Statement createStatement( ) throws
SQLException;
```

เมื่อมีอินแสตนท์ ของ Statement แล้ว เราก็สามารถส่งประโยค SQL ไปสู่ระบบฐานข้อมูล โดยใช้ methods สามแบบของ Statement ดังนี้

```
public abstract ResultSet executeQuery(String) throws
SQLException;
public abstract int executeUpdate(String) throws SQLException;
public abstract boolean execute(String) throws SQLException;
```

ทั้ง 3 method นี้มีพารามิเตอร์ string เป็นประโยค SQL ที่จะถูกส่งไประบบฐานข้อมูล แต่ทั้งสาม methods มีข้อแตกต่างกันดังนี้คือ

เราใช้ executeQuery ( ) ในกรณีที่ประโยค SQL นั้นมีผลลัพธ์กลับมา ซึ่งก็คือประโยค SELECT หรือการเรียกไปที่ stored procedures ที่มีจำนวนอาร์กิวเมนต์คงที่ ผลลัพธ์ที่ระบบฐานข้อมูลส่งกลับมาคือตารางที่อยู่ในรูปของ stream ของ records โดยสร้างเป็น instace ของคลาส ResultSet

ตัวอย่างการใช้ executeUpdate ( ) ทำงานประโยค UPDATE

```
// UpdateTest.java
import java.sql.*;
class UpdateTest
{
      long a=75690191264;
      public static void main(String argv[]) throws Exception
      {

            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            Connection c = DriverManager.getConnection
            ("Jdbc:Odbc:skytrain", "", "");
            Statement s = c.createStatement();
            s.executeUpdate("UPDATE skytrain SET balance = 100 WHERE
            id ="+a  );
            s.close(); c.close();
      }
}
```

โปรแกรมนี้ใช้ executeUpdate ( ) ทำงานประโยค UPDATE เพื่อเปลี่ยน gpa ของ Jame Bond ให้เป็น 3.4 เมื่อทำงานโปรแกรมนี้แล้ว ตรวจสอบผลที่เกิดขึ้นกับการเปลี่ยนแปลงต่อข้อมูลใน student table

2.4 การสร้างจาวาแอปพลิเคชั่น (Java Application)

จาวาแอปพลิเคชั่น คือโปรแกรมที่สร้างขึ้นด้วยภาษาจาวาเพื่อถูกทำงานโดยจาวาอินเทอร์พริเตอร์ (Java interpreter) ต้องเป็นโปรแกรมที่สมบูรณ์ มี main () เป็นจุดเริ่มต้น และสามารถควบคุมการดำเนินไปของตัวเองได้ ทำงานภายใต้ จาวา อินเทอร์พริเตอร์ โดยไม่ต้องมีโปรแกรมอื่นช่วย บางคนจึงเรียกว่าแสตนอะโลนโปรแกรม (stand alone program)

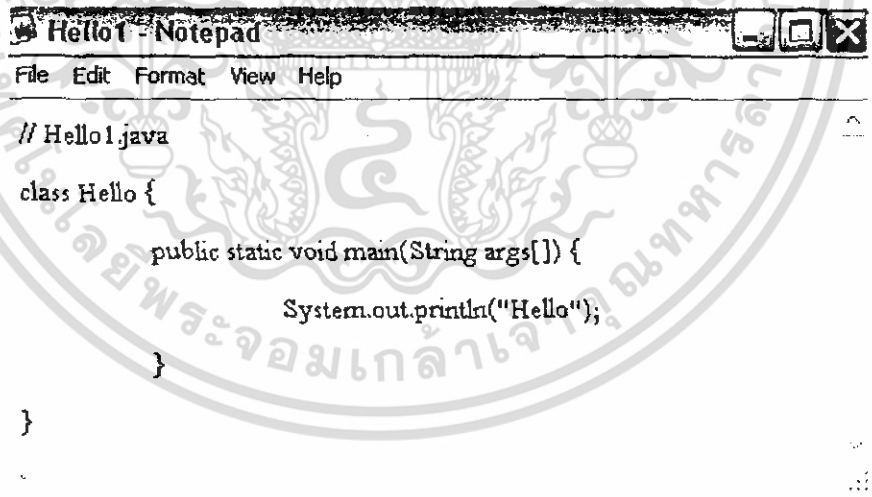
ขั้นตอนการสร้างและทำงาน จาวาแอปพลิเคชั่น มีดังนี้ เริ่มจากใช้ Editor (อย่างเช่น notepad) เขียนโปรแกรมสำหรับ จาวาแอปพลิเคชั่น นั้นเก็บไว้ในไฟล์ที่มีนามสกุลเป็น .java เช่น x.java จากนั้นใช้ javac.exe ทำการคอมไพล์ x.java จะได้ผลลัพธ์เป็นไฟล์ x.class ซึ่งเป็นโปรแกรม JVM ของคลาส X เมื่อต้องการให้ X.class นั้นทำงานก็ส่งให้ java.exe ดังแสดงในรูปนี้

รูปที่ 2.12 แสดงกระบวนการสร้างจาวาแอปพลิเคชั่น

ตัวอย่างการสร้างจาวาแอปพลิเคชั่น ที่พิมพ์คำว่า Hello ออกมาที่จอภาพ

สมมติว่า เราจะทำงานในไดเรกเทอรี C:\MyJava\Ch1 ขั้นตอนต่อไปลำดับได้ดังนี้

1. ใช้ notepad พิมพ์โปรแกรมลงไปดังในรูปด้านล่างนี้ เมื่อเสร็จแล้วบันทึก ลงในไฟล์ Hello1.java ใน C:\

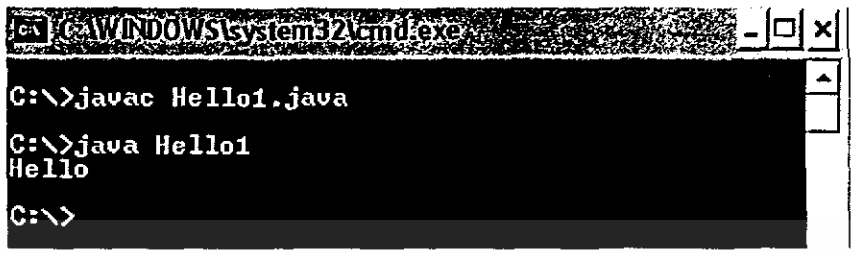


รูปที่ 2.13 แสดงการพิมพ์โปรแกรม

2. เปิด MS-Dos Prompt window และเปลี่ยนไดเรกเทอรี่ไปที่ C:\

3. คอมไพล์โปรแกรม Hello.java โดยพิมพ์คำสั่งที่ command line ใน prompt window ดังนี้ javac Hello1.java จะได้ไฟล์ Hello1.class เป็นผลลัพธ์

4. เรียก java.exe ให้ทำงานโปรแกรม Hello1.class โดยพิมพ์คำสั่งที่ command line ดังนี้ Java Hello1 จะได้ "Hello" พิมพ์ออกไปที่จอภาพขั้นตอนที่ 3 ถึง 4 แสดงในรูปด้านล่างนี้



รูปที่ 2.14 แสดงการคอมไพล์และรันโปรแกรมบนคอมมานไลน์

การใช้งาน Java Compiler (javac.exe) มีข้อสังเกตดังนี้ โปรแกรมภาษาจาวาจะต้องเก็บในไฟล์ที่มี extension เป็น .java เมื่อถูกคอมไพล์แล้วจะได้ไฟล์ที่มีชื่อเหมือนกับคลาสที่กำหนดในโปรแกรมนั้น และมี extension เป็น .class โดยจะได้หนึ่งไฟล์ .class ต่อหนึ่งคลาสที่กำหนดในโปรแกรมนั้น เราสามารถคอมไพล์โปรแกรมภาษาจาวาหลายๆ ไฟล์ในการเรียก javac.exe เพียงครั้งเดียวก็ได้ โดยใช้ wildcard* เช่น หากมีไฟล์ .java หลายๆไฟล์ในไดเรกเทอรีที่กำลังทำงานอยู่ ก็คอมไพล์นั้นทั้งหมดโดย javac *.java

หากโปรแกรมจาวาหนึ่งอ้างถึงคลาสอื่นๆคอมไพลเลอร์ javac.exe จะเริ่มค้นหาคลาสนั้น (ไฟล์ .class ของคลาสนั้น) ในตำแหน่งที่ระบุในคลาสพาร์ธ (class path) ซึ่งเรากำหนดไว้ว่า เริ่มจากไดเรกเทอรีปัจจุบัน หากไม่พบให้ไปหาต่อที่ C:\jdk1.4.2_05\lib\classes.zip แต่หากคอมไพเลอร์พบว่า ในไดเรกทอรีปัจจุบันมีไฟล์ .java ของคลาสที่ถูกอ้างถึงซึ่งยังไม่ถูกคอมไพล์ หรือถูกเปลี่ยนแปลงใหม่กว่าไฟล์ .class ที่ถูกคอมไพล์แล้ว คอมไพเลอร์ก็จะคอมไพล์ไฟล์นั้นให้เองโดยอัตโนมัติ ความสามารถของ javac.exe นี้ ทำให้ภาษา java ไม่จำเป็นต้องมีโปรแกรมช่วยจัดการคอมไพล์ไฟล์หลายๆ ไฟล์อย่างเช่น make หรือ makefile

การใช้งาน Java Interpreter (java.exe) มีข้อสังเกตดังนี้

ไฟล์ .class หนึ่งจะมีเพียงหนึ่งคลาส โดยทั่วไป java.exe จะเริ่มทำงานจากคลาสที่มี main ( ) ซึ่งเป็นจุดเริ่มต้นของโปรแกรม ส่วนคลาสอื่นๆ จะถูกโหลดเข้ามาในอินเทอร์พริเตอร์ เมื่อถูกอ้างอิง ขณะที่โปรแกรมทำงาน

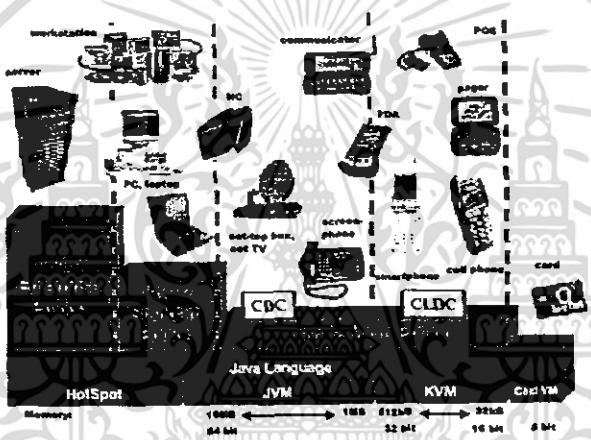
java.exe อาจถูกเรียกจากคอมมานไลน์(command line)หรือจากแบทช์ไฟล์ (batch file) ก็ได้ และอาจมีอากิวเมนต์ (arguments) ส่งให้แก่โปรแกรมไปเป็นอาเรย์ (array) ของสตริง (String) ที่ชื่อเป็น args[ ]

ใน java.exe มีโปรแกรมตรวจสอบไบท์โค้ด (byte code verifier) สำหรับตรวจสอบคลาสที่ถูกโหลดเข้ามาทำงาน หากพบว่ามีการกระทำที่ไม่สมควร ก็ปฏิเสธที่จะทำงานคลาสนั้น โดยดีฟอล์ท (default) ถ้าคลาสนั้นอยู่ในเครื่องเดียวกับ java.exe ที่ทำงาน ก็จะถือว่าเป็นคลาสที่ไว้ใจได้และตัวตรวจสอบความถูกต้องไบท์ จะยกเว้นการตรวจสอบเพื่อความเร็วในการทำงาน แต่หากเป็นคลาสที่โหลดมาจากเครื่องอื่นถือว่าเป็นคลาสที่ไว้ใจไม่ได้และต้องทำการตรวจสอบ

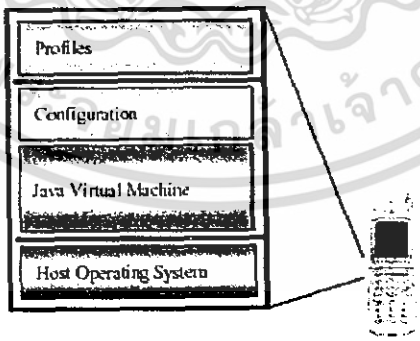
2.5 J2ME (Java 2 Platform Micro Edition)

J2ME จะใช้ในการพัฒนาแอปพลิเคชั่นจาวาบนอุปกรณ์มือถือ, เครื่องปาล์ม, พีดีเอ และ อุปกรณ์อื่นๆ ที่มีหน่วยความจำที่ไม่มากนัก โดยที่อุปกรณ์เหล่านั้นจะต้องรองรับกับ J2ME ซึ่งในปัจจุบันก็มีกันอยู่มากมาย J2ME จะมีการแบ่งการพัฒนาออกเป็น 2 ส่วนอันได้แก่ อุปกรณ์ที่มีทรัพยากรพอสมควร คือตั้งแต่ 1 – 10 เมกกะไบท์จะใช้ชุดคำสั่งของ CDC (Connected Device Configuration) ในการพัฒนา แต่สำหรับอุปกรณ์ที่มี ทรัพยากรน้อย อย่างเช่น โทรศัพท์มือถือ (Smartphone, Cell phone) ก็จะใช้ชุดคำสั่งที่เป็น CLDC (Connected Limited Device Configuration) ในการพัฒนา และจะทำงานบน KVM (K Virtual Machine)

โครงสร้างของ J2ME



รูปที่ 2.15 แสดงการแบ่งแพลตฟอร์มต่างๆของภาษาจาวา



รูปที่ 2.16 แสดงโปรแกรมเลเยอร์แสตก (Software Layer Stack)

จากรูปที่ 2.16 เป็นรูปของ J2ME เลเยอร์แสตกซึ่งจะเห็นได้ว่ามีอยู่ 4 ชั้นด้วยกันอันได้แก่ โฮสโอเปอร์เรติ้งซิสเต็ม (Host Operating System) ซึ่งทำหน้าที่เป็นระบบปฏิบัติการของมือถือ, จาวาเวอร์ชัวแมชชีน (Java virtual machine) จะเป็นสิ่งแวดล้อมสำหรับการรันโปรแกรม, คอนฟิกูเรชั่น (Configuration) จะเป็นชุด

ของ คลาสไลเบอรี่ (Class Library) ต่างๆสำหรับอุปกรณ์ ซึ่งสำหรับอุปกรณ์ที่มีทรัพยากรน้อยจะใช้ชุดคำสั่ง CLDC (Connected Limited Device Configuration) และสำหรับชั้นสุดท้ายที่ชื่อโพรไฟล์จะเป็นชุดคำสั่งที่เน้นถึงอุปกรณ์แต่ละชนิด เช่น MIDP เป็นโพรไฟล์สำหรับโทรศัพท์มือถือ

เริ่มการติดตั้งโปรแกรม

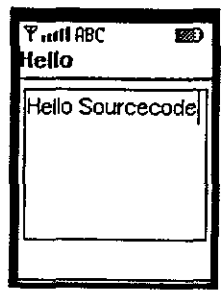
1) เริ่มต้นการติดตั้งโปรแกรมโดยเริ่มที่การติดตั้ง J2SE
2) หลังจากนั้นก็ให้ทำการดาวน์โหลดโปรแกรม J2ME Wireless Toolkit
3) เมื่อติดตั้งถึงขั้นตอนของการเลือกพาร์ธของจาวาเวอร์ชัวแมชชีนแล้ว ให้ทำการเลือก (Browse) ไปที่พาร์ธ J2SE ที่ได้ทำการลงไว้ก่อนหน้า หลังจากนั้นก็คลิก Next เพื่อไปขั้นตอนอื่นๆ ต่อไปเรื่อยๆ
4) เมื่อถึงขั้นตอน Setup Type ให้เลือก Stand Alone แล้วคลิกปุ่ม Next ไปจนสิ้นสุดการติดตั้ง

ตัวอย่างการสร้างมิดเลท (MIDLET)

1) เริ่มต้นการสร้างโปรแกรมโดยเรียกใช้โปรแกรมจาก เมนู Start->Programs->J2ME Wireless Toolkit 2.2 -- > KToolBar
2) คลิกที่ปุ่ม New Project แล้วทำการป้อนค่า Project Name เป็น Hello และ MIDlet Class Name เป็น HelloSC คลิกปุ่ม Create Project เพื่อสร้าง Project ดังแสดงตามรูปด้านล่าง

รูปที่ 2.17 แสดงการสร้างมิดเลท (Midlet)

3) หลังจากการคลิกสร้าง Project แล้ว จะปรากฏหน้าต่างที่แสดงถึงคุณสมบัติต่างๆของโปรเจ็ค ที่เราได้ทำการสร้าง ต่อไปจะเป็นการสร้างส่วนของโค้ดโดยเปิดโปรแกรม NotePad แล้วทำการป้อนโค้ด ดังแสดงในตารางด้านล่าง เสร็จแล้วให้ ทำการบันทึกไฟล์ในชื่อ HelloSC.java ไว้ตามพาร์ธ C:\J2mewtk\apps\Hello\src (ในที่นี้ได้ทำการลงโปรแกรม ไว้ที่ Drive C ซึ่งหากลงที่ Drive อื่น พาร์ธก็จะเปลี่ยนไปด้วย
4) ทำการคลิกปุ่ม Build ที่หน้าต่างของโปรแกรม KToolBar, เมื่อ Build เสร็จแล้วให้ทำการรันโปรแกรม โดยคลิกที่ปุ่ม จะปรากฏหน้าต่างที่จำลองการทำงานของเครื่องโทรศัพท์มือถือ ที่รันโปรแกรมที่เราเขียนขึ้นมา ดังรูปจะแสดงผลที่ได้จากการเขียนโค้ดข้างต้น



รูปที่ 2.18 แสดงหน้าจอของอีมูเลเตอร์หลังจากรันโปรแกรม

บทที่ 3
การออกแบบและขั้นตอนการทำงาน

3.1 การออกแบบ

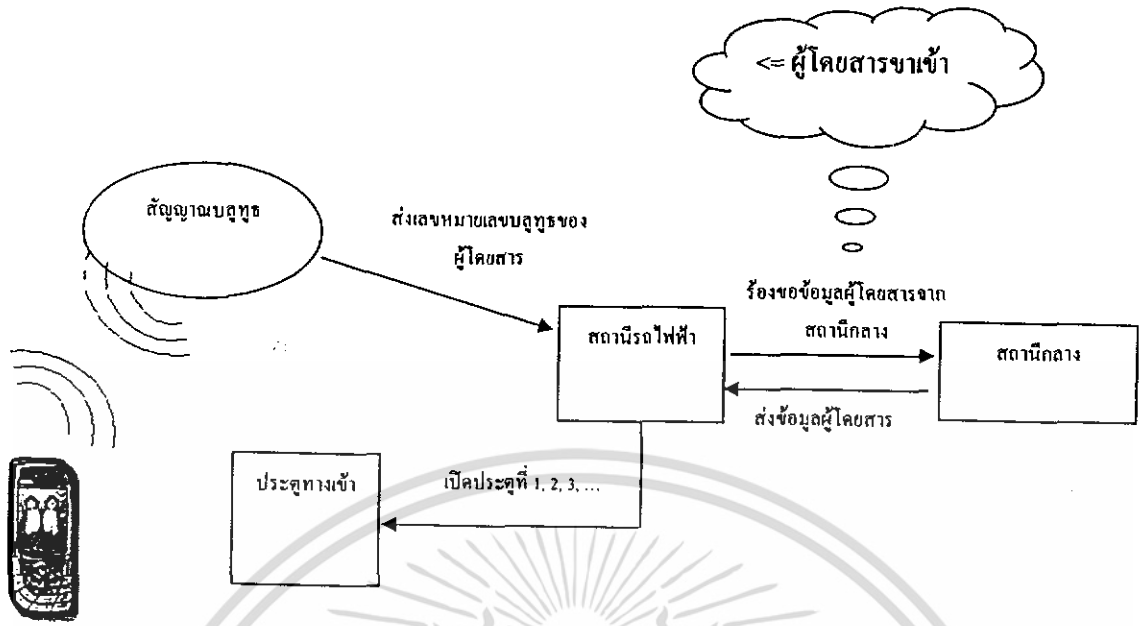
โครงงานนี้ได้ออกแบบการทำงาน 3 ส่วนหลักได้แก่ ส่วนของโปรแกรมที่รันอยู่บนโทรศัพท์มือถือของผู้โดยสาร ส่วนของโปรแกรมที่รันอยู่บนสถานีย่อย และส่วนของโปรแกรมที่รันอยู่บนสถานีกลาง นอกจากนั้นยังมีส่วนของฐานข้อมูลที่เก็บอยู่ที่สถานีกลาง

เทคโนโลยีหลักที่ใช้ในการสร้างโครงงานชิ้นนี้ได้แก่

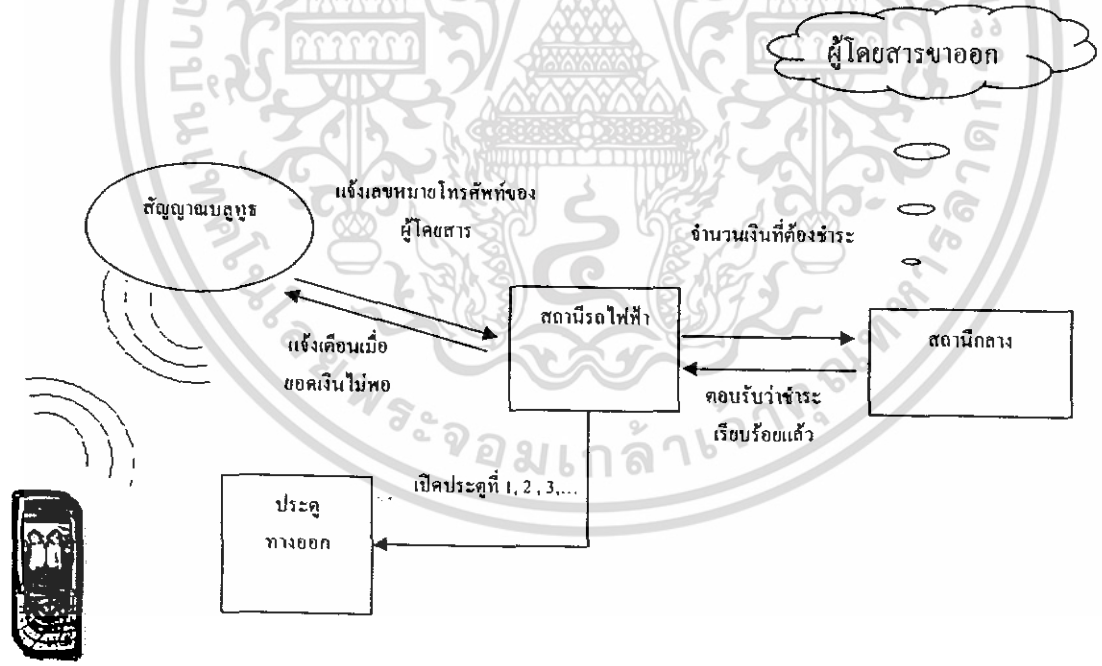
1. Java Bluetooth
2. Java Networking
3. Java Data Base Connection (JDBC)
4. J2ME

3.2 ขั้นตอนการทำงาน

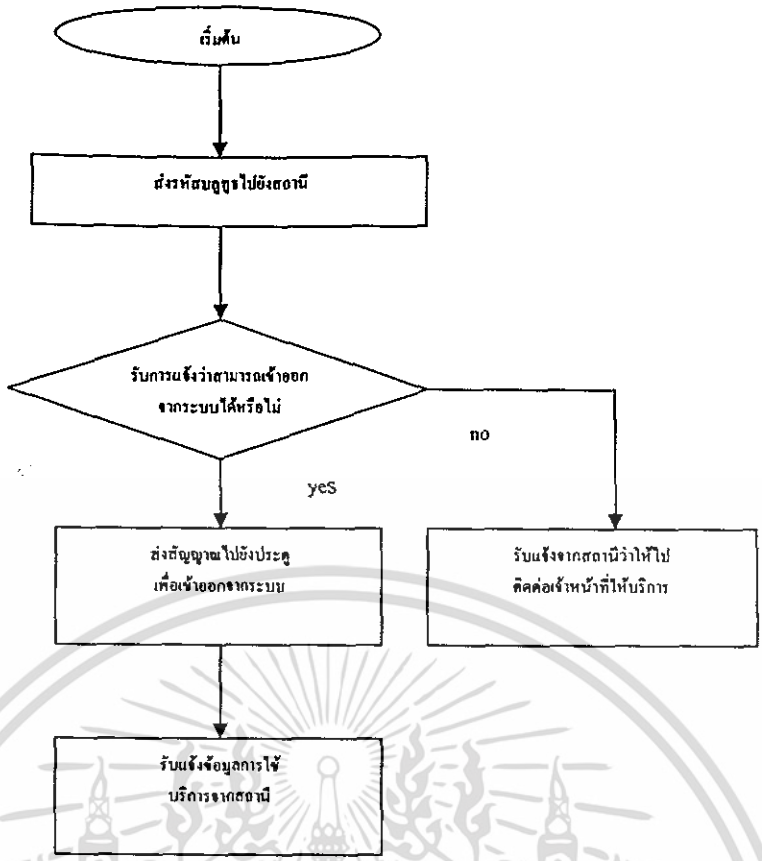
1) ผู้โดยสารต้องเปิดแอปพลิเคชั่นการชำระค่าโดยสารรถไฟฟ้าและเข้าออกระบบรถไฟฟ้าผ่านระบบโทรศัพท์มือถือ
2) เมื่อผู้โดยสารอยู่ในบริเวณสถานีรถไฟฟ้าโทรศัพท์มือถือจะทำการเชื่อมต่อกับสถานีย่อย (station server) โดยใช้สัญญาณบลูทูธในการเชื่อมต่อและรับส่งข้อมูลระหว่างโทรศัพท์มือถือกับสถานี
3) สถานีย่อยจะรับรหัสบลูทูธของโทรศัพท์มือถือผู้โดยสารและส่งรหัสบลูทูธต่อไปยังสถานีกลาง (main server) สถานีกลางจะใช้รหัสบลูทูธในการดูข้อมูลผู้โดยสารที่อยู่ในฐานข้อมูล ประมวลผลตรวจสอบว่าเป็นการเข้าหรือออกจากระบบ คำนวณค่าใช้จ่ายการเดินทาง และเป็นตัวส่งสัญญาณอนุญาตให้ผู้โดยสารสามารถเข้าออกจากระบบต่อไปยังสถานีย่อยที่ผู้โดยสารอยู่
4) หากผู้โดยสารสามารถที่จะชำระค่าบริการ สถานีย่อยซึ่งได้รับสัญญาณอนุญาตให้ผู้โดยสารสามารถเข้าออกจากระบบได้จากสถานีกลาง จะส่งสัญญาณไปยังประตูทางเข้าเพื่อให้รอรับสัญญาณการเปิดประตูจากผู้โดยสารที่อยู่ข้างหน้าประตู
5) ผู้โดยสารผ่านประตู ผู้โดยสารจะได้รับแจ้งทั้งขาเข้าและขาออกจากระบบถึงยอดเงินคงเหลือ และค่าเดินทาง
6) หากผู้โดยสารไม่สามารถที่จะชำระค่าบริการ ก็จะไม่มีสัญญาณอนุญาตให้ประตูเปิดจากสถานี ผู้โดยสารจะไม่สามารถผ่านประตูได้และจะได้รับข้อความแจ้งให้รับทราบในกรณีที่ไม่สามารถเข้าออกระบบรวมถึงวิธีการแก้ไข



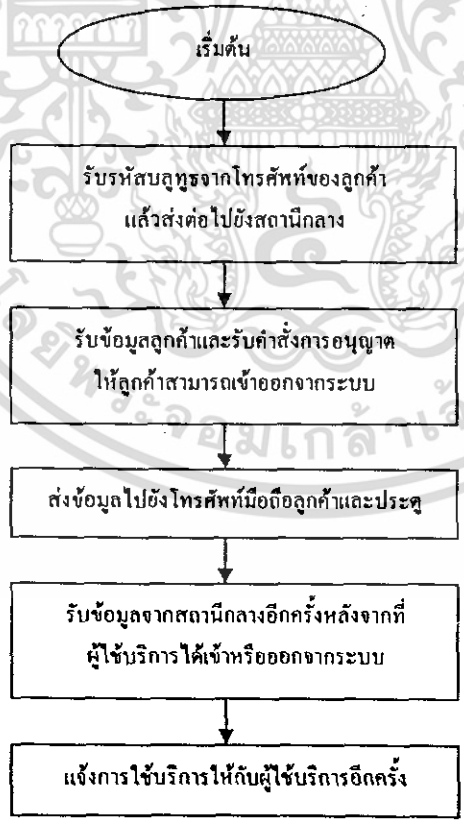
รูปที่ 3.1 แสดงขั้นตอนการเข้าสู่ระบบ



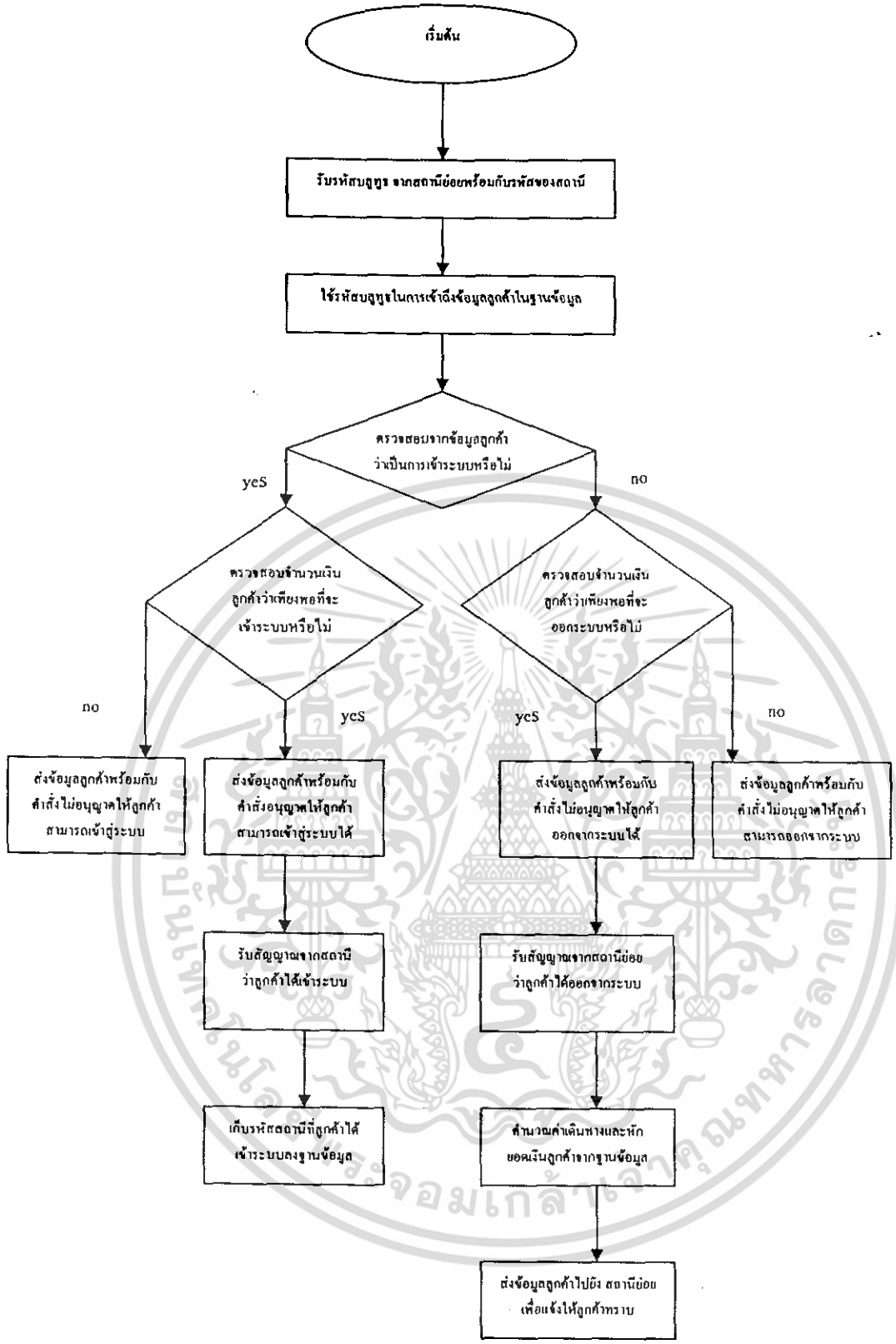
รูปที่ 3.2 แสดงขั้นตอนการออกจากระบบ



รูปที่ 3.3 แสดงโฟลว์ชาร์ทของการทำงานของโทรศัพท์มือถือ



รูปที่ 3.4 แสดงโฟลว์ชาร์ทไดอาแกรมการทำงานของสถานีย่อย



รูปที่ 3.5 แสดงโฟลว์ชาร์ทไดอาแกรมการทำงานของสถานีกลาง

บทที่ 4

การทดลองและผลการทดลอง

4.1 สร้างฐานข้อมูล

เริ่มทดลองโดยสร้างและกำหนดค่าสมมติต่างๆลงไปฐานข้อมูล

4.1.1 ฐานข้อมูลของผู้โดยสาร

ฐานข้อมูลของผู้โดยสารจะมีอยู่ 2 คอลัมน์คือ

- Id เป็นเลขรหัสบลูทูธจำนวน 11 หลักของโทรศัพท์มือถือผู้โดยสาร โดยเลขรหัสบลูทูธของโทรศัพท์แต่ละเครื่องจะไม่ซ้ำกัน
- balance เป็นยอดเงินค่าบริการรถไฟฟ้าคงเหลือของแต่ละโทรศัพท์แต่ละเครื่อง

	Id	balance
	23819836789	500
	75690191264	25
	75695795761	200
	89928364810	1000
	23789915678	290
	34782349723	450

รูปที่ 4.1 แสดงฐานข้อมูลของผู้โดยสาร

4.1.2 ฐานข้อมูลของสถานี

ฐานข้อมูลสถานีย่อยจะมีอยู่ด้วยกัน 4 คอลัมน์คือ

- อินเด็กซ์เปรียบเสมือนว่าเป็นรหัสของแต่ละสถานีย่อย ไว้เพื่อระบุว่าเป็นสถานีใด
- ชื่อของสถานี
- ระยะห่างนับจากศูนย์กลางซึ่งก็คือสถานีสยาม

4.1.3 ฐานข้อมูลค่าใช้จ่าย

ฐานข้อมูลค่าใช้จ่ายมีอยู่ด้วยกัน 2 คอลัมน์คือ

- ระยะทางที่เดินทางโดยกำหนดเป็นโซน
- ค่าใช้จ่ายในการเดินทางไปแต่ละโซน

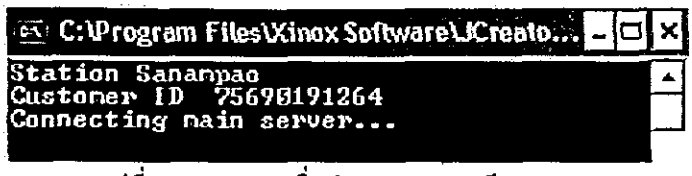
Index	station	distance	direction
1	Mo Chit	7	1
2	Saphan Khwai	6	1
3	Ari	5	1
4	Sanam Pao	4	1
5	Victory Monumen	3	1
6	Phaya Thai	2	1
7	Ratchathewi	1	1
8	Siam	0	0
9	Chit Lom	1	2
10	Phloen Chit	2	2
11	Nana	3	2
12	Asok	4	2
13	Phrom Phong	5	2
14	Thong Lo	6	2
15	Ekkamai	7	2
16	Phra Khanong	8	2
17	On nut	9	2
18	National Stadium	1	3
19	Ratchadumri	1	4
20	Saladaeng	2	4
21	Chongnonsri	3	4
22	Surasuk	4	4

รูปที่ 4.2 แสดงฐานข้อมูลของสถานีย่อย

distance	fare
1	10
2	15
3	20
4	25
5	30
6	35
7	40
8	40
9	40
10	40
11	40
12	40
13	40
14	40
15	40
16	40
17	40

รูปที่ 4.3 แสดงฐานข้อมูลของค่าใช้จ่ายในการเดินทางไปแต่ละโซน

4.2 รันเซอร์เวอร์สถานี



รูปที่ 4.4 แสดงการเริ่มทำงานของสถานี

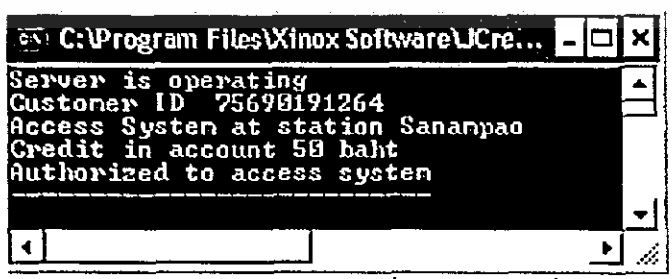
4.3 รันเซอร์เวอร์สถานีกลาง

รูปที่ 4.5 แสดงการเริ่มทำงานของสถานีกลาง

4.4 ทดสอบ

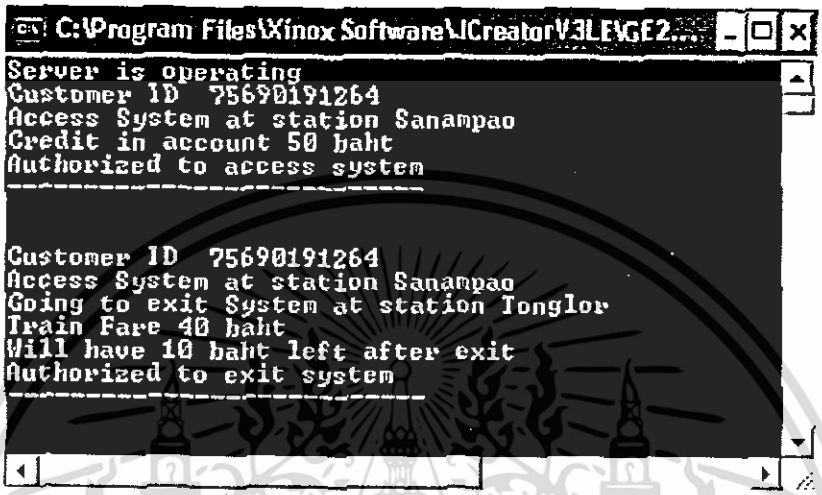
ทำได้โดยนำโทรศัพท์มือถือที่มีหมายเลขบลูทูธ 75690191264 เข้าใกล้เครื่องคอมพิวเตอร์ที่ทำหน้าที่เป็นสถานีต่างๆ จะพบว่าที่คอมพิวเตอร์อีกเครื่องที่เป็นสถานีกลางจะแสดงหมายเลขบลูทูธที่ตรงกัน ชื่อสถานีที่เข้าและออก คำสั่งอนุญาตให้เข้าสู่ระบบ และยอดเงินคงเหลือของผู้โดยสาร ยอดเงินคงเหลือในฐานข้อมูล จะเปลี่ยนแปลงทันทีที่ผู้โดยสารได้เข้าและออกจากระบบ

ในการทำงานจะมีเงื่อนไข 3 กรณี ก็คือ
กรณีที่ 1 เมื่อผู้โดยสารมีจำนวนเงินคงเหลือในบัญชีมากเพียงพอที่จะเข้าและออกจากระบบ การทำงานของสถานีกลางจะแสดงดังรูปภาพต่อไปนี้



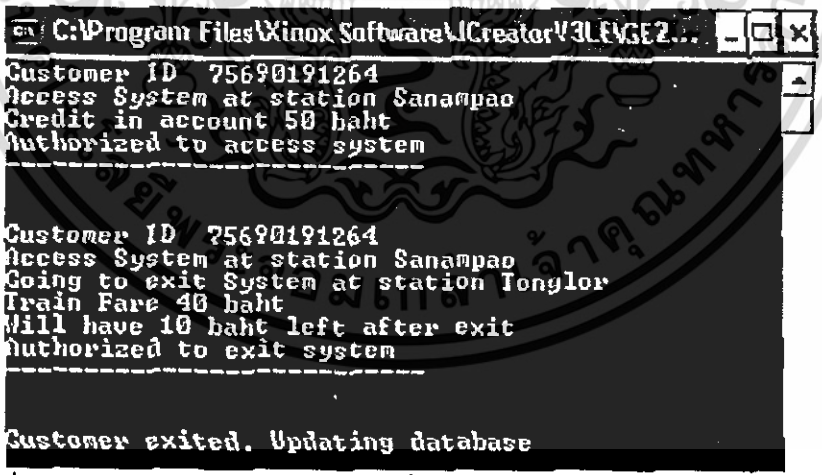
รูปที่ 4.6 แสดงการทำงานของสถานีกลางเมื่อผู้โดยสารอยู่ที่สถานีสนามเป้า

- กดปุ่มเปิดประตูจากโทรศัพท์มือถือ
- รันสถานีสนามเป้าเปรียบเสมือนว่าผู้โดยสารได้เดินทางถึงสถานีสนามเป้าเรียบร้อยแล้ว สถานีสนามเป้าจะทำการเชื่อมต่อกับสถานีกลาง สถานีกลางจะประมวลผลคิดอัตราค่าโดยสารและส่งกลับมายังสถานีสนามเป้า



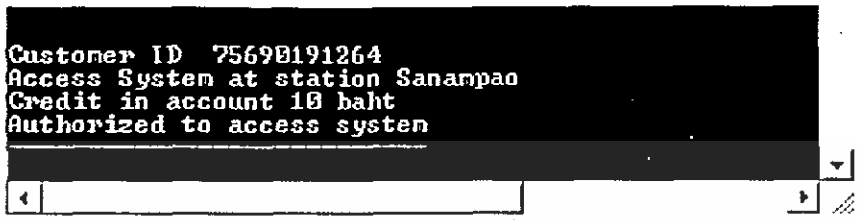
รูปที่ 4.7 แสดงการทำงานของสถานีกลางเมื่อผู้โดยสารกำลังจะออกจากสถานีทองหล่อ

- หลังจากนั้นกดปุ่มเปิดประตูที่โทรศัพท์มือถืออีกครั้ง
- สถานีกลางจะทำการอัพเดทฐานข้อมูล



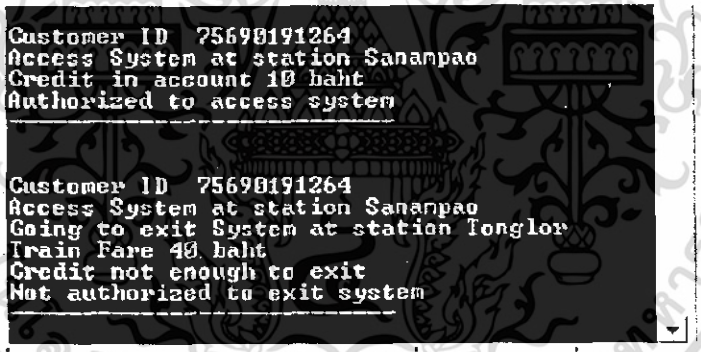
รูปที่ 4.8 แสดงการทำงานของสถานีกลางเมื่อผู้โดยสารได้ผ่านประตูออกจากสถานีทองหล่อแล้ว

กรณีที่ 2 เมื่อผู้โดยสารมีจำนวนเงินคงเหลือในบัญชีมากเพียงพอที่จะเข้าสู่ระบบแต่ไม่เพียงพอที่จะออกจากระบบ การทำงานของสถานีกลางจะแสดงดังรูปภาพต่อไปนี้



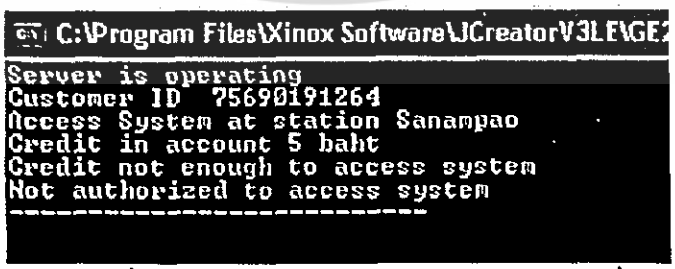
รูปที่ 4.9 แสดงการทำงานของสถานีกลางเมื่อผู้โดยสารอยู่ที่สถานีสนามเป้า และมีเงินคงเหลือในบัญชีขั้นต่ำคือ 10 บาท เพียงพอที่จะเข้าสู่ระบบ

- กดปุ่มเปิดประตูจากโทรศัพท์มือถือ
- รันสถานีสนามเป้าเปรียบเสมือนว่าผู้โดยสารได้เดินทางถึงสถานีสนามเป้าเรียบร้อยแล้ว สถานีสนามเป้าจะทำการเชื่อมต่อกับสถานีกลาง สถานีกลางจะประมวลผลปรากฏว่ามียอดเงินไม่เพียงพอที่จะออกจากระบบ จึงให้สัญญาณการอนุญาตกลับไปยังสถานี

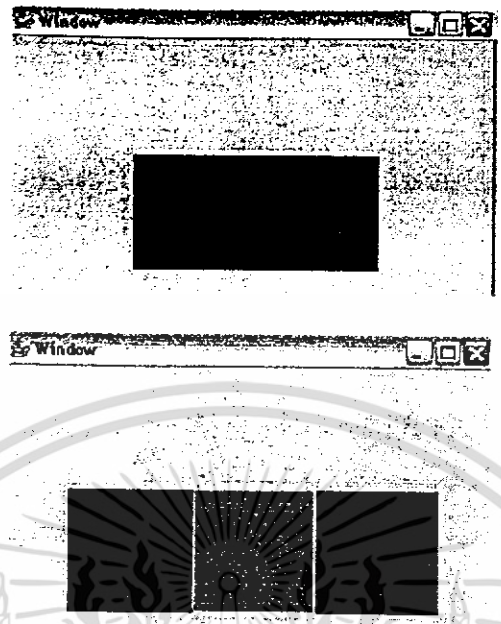


รูปที่ 4.10 แสดงการทำงานของสถานีกลางเมื่อผู้โดยสารอยู่ที่สถานีทองหล่อ มีเงินคงเหลือในบัญชีไม่เพียงพอที่จะออกจากระบบ

กรณีที่ 3 เมื่อลูกค้ามีเงินคงเหลือในบัญชีไม่ถึงขั้นต่ำในการเข้าสู่ระบบ



รูปที่ 4.11 แสดงการทำงานของสถานีกลาง ในกรณีที่ลูกค้ามีเงินคงเหลือในบัญชีไม่ถึงขั้นต่ำในการเข้าสู่ระบบ



รูปที่ 4.12 แสดงรูปประตูจำลองทางเข้าออกของสถานี

บทที่ 5
บทวิจารณ์และบทสรุป

5.1 สรุปการทำโครงงาน

- ความรู้ที่ได้รับจากการทำโครงงานนี้ได้แก่

1. เรียนรู้การเขียนโปรแกรมเพื่อเชื่อมต่อโทรศัพท์มือถือกับเครื่องคอมพิวเตอร์ด้วยภาษาจาวา โดยใช้บลูทูธเชื่อมต่อกัน และใช้ api J2R-82
2. เรียนรู้การสร้างเครือข่ายเน็ตเวิร์ค และการติดต่อสื่อสารระหว่างเซอฟเวอร์กับไคลแอนท์
3. เรียนรู้การสร้างและออกแบบและเชื่อมต่อฐานข้อมูล
4. เรียนรู้การสร้างกราฟิกอินเตอร์เฟส

- ปัญหาที่ได้รับในการทำโครงงานชิ้นนี้

เนื่องจากบลูทูธค่อนข้างเป็นเทคโนโลยีใหม่จึงหาแหล่งข้อมูลในการทำงานได้น้อย รวมถึงโทรศัพท์ที่จะนำมาทดลองค่อนข้างมีราคาสูง

5.2 วิจารณ์โครงงาน

โครงงานนี้เป็นแอปพลิเคชั่นที่มีขนาดใหญ่และเกี่ยวข้องกับระดับองค์กรรถไฟฟ้า แนวทางในการพัฒนาต่อไปจะต้องพัฒนาทางด้านการรับส่งสัญญาณระหว่างฮาร์ดแวร์จริงๆซึ่งก็คือประตูทางเข้าออก ระบบการตรวจสอบและการรักษาความปลอดภัยของข้อมูลโดยเฉพาะยอดเงินของผู้ใช้บริการ ระบบการเพิ่มจำนวนยอดเงิน รวมทั้งรายละเอียดปลีกย่อยอีกมากมาย

ในการพัฒนาต่อไปจนสามารถใช้งานจริงในอนาคตจึงมีความจำเป็นที่จะต้องปรึกษาหารือและขอความร่วมมือกับองค์กรรถไฟฟ้า

บรรณานุกรม

1. เก่ง J2ME ให้ครบสูตร

   ทรงเกียรติ์ ภาวดี, สำนักพิมพ์ บริษัท วิตี้กรุ๊ป จำกัด, จัดจำหน่ายโดย ซีเอ็ดยูเคชั่น จำกัด (มหาชน)

2. การเขียนโปรแกรมสำหรับ Wireless Application ด้วย J2ME

   พ.อ. เจณวิทย์ เหลืองอร่าม, ปิยวิทย์ เหลืองอร่าม, สำนักพิมพ์บริษัทธรรมสาร จำกัด

   จัดจำหน่ายโดย ซีเอ็ดยูเคชั่น จำกัด (มหาชน)

3. Java Programming Volume 3

   ดร.วีรศักดิ์ ชึงถาวร

   จัดจำหน่ายโดย ซีเอ็ดยูเคชั่น จำกัด (มหาชน)

4. Java How to Program fith Edition

   H.M. Deitel, P.J. Deitel, Prentice Hall, USR, NJ07458

5. Programming Java 2 Micro Edition on Symbian OS

   Martin de Jode

   Published by John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester,

   West Sussex PO19 8SQ, England

   http://java.sun.com

6. http://java.sun.com

ภาคผนวก

Station server

BTServicAndDeviceAgent

```
import java.io.*;
import javax.bluetooth.*;
import com.atinav.standardedition.io.*;

public final class BTServicAndDeviceAgent
{
      boolean running2=true;
      boolean conn2;
      private String UUID_STRING;
      private String SERVICE_NAME;
      private LocalDevice localDevice;
      private DiscoveryAgent agent;
      private L2CAPConnection conn;
      private RemoteDevice[] remoteDevices;
      private boolean running = false;
      private String msg;
      private String[] balance;
      private BTServicAndDeviceDiscovery discoverer;
      private String[] devices;
      public String getId,String1,String2,Authorize;
      private String[] remoteId;
      private long bluetoothaddress;
      StationServer stationserver = new StationServer();
      private StartSystem startsystem;
      private BTServicAndDeviceAgent(){ }
      public  BTServicAndDeviceAgent(String UUID_STRING,String
      SERVICE_NAME,StartSystem startsystem)
      {
            this.UUID_STRING = UUID_STRING;
            this.SERVICE_NAME = SERVICE_NAME;
            this.startsystem=startsystem;
            discoverer = new
            BTServicAndDeviceDiscovery(SERVICE_NAME,this);
            try
            {
            localDevice = LocalDevice.getLocalDevice();
            System.out.println(localDevice.getBluetoothAddress());
            agent = localDevice.getDiscoveryAgent();
            }
            catch(BluetoothStateException bse)
            {
                  bse.printStackTrace();
            }
      }

      //actionPerformed-StartServer
      public void StartServer() {
         // ToSendData(String1);
            new Thread() {
                  public void run() {
                        try{

      localDevice.setDiscoverable(DiscoveryAgent.GIAC);
      L2CAPConnectionNotifier notifier =
(L2CAPConnectionNotifier)Connector.open("btl2cap://localhost:" +
UUID_STRING + ";name=" + SERVICE_NAME);
      ServiceRecord record = localDevice.getRecord(notifier);
```

```
String connURL =
record.getConnectionURL(ServiceRecord.NOAUTHENTICATE_NOENCRYPT,
false);
      //    Waits for a client to connect to this L2CAPServer
conn = notifier.acceptAndOpen();
new Thread()
{
      public void run()
      {
            running = true;
            ToReciveData();
      }
}.start();
                  } catch(IOException ioe) {
                        ioe.printStackTrace();
                  }
            }
      }.start();
}

public void StopServer() {
      running = false;
}

public void SearchDevice()
{
      try {
        agent.startInquiry(DiscoveryAgent.GIAC, discoverer);
          }
       catch(BluetoothStateException bse){
        bse.printStackTrace();
          }
}

public void SearchService(int index)
{
      int[] attrSet = {0x0100};//return service name attribute
      UUID[] uuidSet = new UUID[1];
      uuidSet[0] = new UUID(UUID_STRING, false);
      try{
      agent.searchServices(attrSet, uuidSet,
      remoteDevices[index], discoverer);
         }
      catch(BluetoothStateException bse)
      {
            bse.printStackTrace();
      }
}

public void StopClient()
{
      running = false;
}

public void deviceInquiryFinished(RemoteDevice[] remoteDevices,
String message)
{
      this.remoteDevices = remoteDevices;
      devices = new String[remoteDevices.length];
      for(int i = 0; i < remoteDevices.length; i++)
      {
```

```
                        try
                        {

                        getId=remoteDevices[i].getBluetoothAddress();
                        bluetoothaddress=Long.parseLong(getId,16);
                        balance=new String[20];
          balance[i]=""+stationserver.returnbalance(bluetoothaddress);
                        String1=stationserver.String1;
                        Authorize=stationserver.Authorize;
                        this.ToSendData(String1);
                        }
                        catch(Exception ioe)
                        {
                         ioe.printStackTrace();
                        }
                    }
                        startsystem.FoundDevice(devices);
      }
      //BTServicAndDeviceDiscovery-serviceSearchCompleted
      public void serviceSearchFinished(ServiceRecord serviceRecord,
String message)
      {
            String url =
serviceRecord.getConnectionURL(ServiceRecord.NOAUTHENTICATE_NOENCRYPT
, false);
            try{
                  conn = (L2CAPConnection)Connector.open(url);
                  new Thread(){
                  public void run()
                  {
                        running = true;
                        ToReciveData();
                  }
      }
}
```

BTServicAndDeviceDiscovery

```
import javax.bluetooth.*;
import java.util.*;
import java.io.*;

public final class BTServicAndDeviceDiscovery implements
DiscoveryListener
{
private Vector remoteDevices = new Vector();
      private ServiceRecord serviceRecord;
      private BTServicAndDeviceAgent callback;
      private String SERVICE_NAME;

      /*From StartSystem-BTServicAndDeviceAgent
      SERVICE_NAME = "L2CAPchat"
      */
      public BTServicAndDeviceDiscovery(String
SERVICE_NAME,BTServicAndDeviceAgent callback)
      {
            this.SERVICE_NAME = SERVICE_NAME;
            this.callback = callback;
      }

      //From where starts the methods?
```

```
      //Called when service(s) are found during a service search

      public void servicesDiscovered(int transID, ServiceRecord[]
servRecord)
      {
            for(int i = 0; i < servRecord.length; i++)
            {
                              DataElement serviceNameElement =
servRecord[i].getAttributeValue(0x0100);//get the Service Name
                              String serviceName =
(String)serviceNameElement.getValue();
                              if(serviceName.equals(SERVICE_NAME))
                              {
                                    serviceRecord = servRecord[0];
                        }
                  }
      }

    // Called when a service search is completed or was terminated
because of an error

    public void serviceSearchCompleted(int transID, int respCode)

    {
            String message = null;

        if (respCode ==
DiscoveryListener.SERVICE_SEARCH_DEVICE_NOT_REACHABLE)
        {
            message = "Device not reachable";
        }
        else if (respCode ==
DiscoveryListener.SERVICE_SEARCH_NO_RECORDS)
        {
                message = "Service not available";
        }
        else if (respCode ==
DiscoveryListener.SERVICE_SEARCH_COMPLETED)
        {
                message = "Service search completed";
        }
        else if (respCode ==
DiscoveryListener.SERVICE_SEARCH_TERMINATED)
        {
                message = "Service search terminated";
        }
        else if (respCode == DiscoveryListener.SERVICE_SEARCH_ERROR)
        {
                message = "Service search error";
        }

        callback.serviceSearchFinished(serviceRecord, message);
}

            //Called when an inquiry is completed

public void inquiryCompleted(int discType)
{
String message = null;

            if (discType == INQUIRY_COMPLETED)
```

```
        {
            message = "Inquiry completed";
        }
        else if (discType == INQUIRY_TERMINATED)
        {
            message = "Inquiry terminated";
        }
        else if (discType == INQUIRY_ERROR)
        {
            message = "Inquiry error";
        }

        RemoteDevice[] devices = new
RemoteDevice[remoteDevices.size()];
        for(int i = 0; i < remoteDevices.size(); i++)
        {
                devices[i] =
(RemoteDevice)remoteDevices.elementAt(i);
            System.out.println(devices[i].getBluetoothAddress());
        }
      callback.deviceInquiryFinished(devices, message);
}

//Called when a device is found during an inquiry
public void deviceDiscovered(RemoteDevice btDevice, DeviceClass cod)
{
remoteDevices.addElement(btDevice);
}

}
```

OpenDoor

```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
import java.lang.*;

public class OpenDoor
{

      PaintPane paintPane;
      JButton closeDoor;
      JButton openDoor;

      int x1_corner = 200;
      int x2_corner = 200;

      public OpenDoor()
      {
      Gui();
      //  ListenEvent();
      } // end constructor

      public void Gui()
      {
            JFrame frame = new JFrame("Window");     // panel
            JPanel first_panel = new JPanel(new BorderLayout());
            JPanel second_panel = new JPanel();     // button
            openDoor = new JButton("openDoor");
```

```
            closeDoor = new JButton("closeDoor");
            second_panel.add(openDoor);
            second_panel.add(closeDoor);
            paintPane = new PaintPane();
            paintPane.setValue(x1_corner,x2_corner);
            first_panel.add(paintPane,BorderLayout.CENTER);
            first_panel.add(second_panel,BorderLayout.SOUTH);
            frame.setSize(800,600);
            frame.setContentPane(first_panel);
            frame.show();
    } // end Gui

public void Open()
{
            if(x1_corner!=100)
            {
                  x1_corner -= 100;
            }
            if(x2_corner!=300)
            {
                  x2_corner += 100;
            }
            paintPane.setValue(x1_corner,x2_corner);
            paintPane.repaint();

} // end Open function
public void Close()
{

            if(x1_corner==100)
            {
                  x1_corner += 100;
            }
            if (x2_corner ==300)
            {
                  x2_corner -= 100;
            }
            paintPane.setValue(x1_corner, x2_corner);
            paintPane.repaint();


} // end close function


public void ListenEvent()
{
    openDoor.addActionListener(new ActionListener()
    {
       public void actionPerformed(ActionEvent e)
       {
           Open();
       }
    });

    closeDoor.addActionListener(new ActionListener()
    {
        public void actionPerformed(ActionEvent e)
        {
           Close();
        }
    });
```

```
 } // end function ListenEvent
}// end of class
```

PaintPane

```
class PaintPane extends JPanel
{

     private int x1_top_corner = 200;
     private int x2_top_corner = 200;
     private int y_top_corner = 200;
     private int width = 200;
     private int height = 200;

     public void setValue(int x1,int x2)
     {
          this.x1_top_corner = x1;
          this.x2_top_corner = x2;
     } // end constructure
     public void show()
     {
           repaint();
     } // end show

     public void paintComponent(Graphics g)
     {
           super.paintComponent(g);

           // set blackground color
            g.setColor(Color.lightGray);
           // blackground
           g.fill3DRect(300, y_top_corner, width, height,false);

           // set color of image
            g.setColor(Color.blue);
           // Door 1 *************** show fill Rect
g.fill3DRect(x1_top_corner, y_top_corner, width, height, true);
           // Door 2 *************** show fill Rect
g.fill3DRect(width+x2_top_corner, y_top_corner, width, height, true);

     } // end paint
} // end class PaintPane
```

StationServer

```
import java.io.*;
import java.net.*;
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
import java.lang.*;

public class StationServer
{


      public String String1,String2,Authorize;
      private static Long LongBalance;
```

```
        public  Long returnbalance(long bluetoothaddress) throws
        Exception
        {

                    Socket s = new Socket("161.246.18.70", 12345);

                    PrintStream op = new
                    PrintStream(s.getOutputStream());
                    op.println(bluetoothaddress);
                    op.println("8");
                    BufferedReader br = new BufferedReader(
                    new InputStreamReader (s.getInputStream()));
                    String1=br.readLine();
                    Authorize=br.readLine();
                    br.close();
                    s.close();
                    System.out.println(String1);
                    System.out.println(Authorize);
                    return LongBalance;
        }
}
```

StartSystem

```
import javax.bluetooth.*;
import java.io.*;
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;

public class StartSystem {
      private static final String UUID_STRING =
      "11112233445566778899AABBCCDDEEFF";
      private static final String UUID_SERVICE = "L2CAPchat";
      OpenDoor opendoor = new OpenDoor();
      private BTServicAndDeviceAgent agent;
      private static Long balance;
      private String id;
      public StartSystem()

      {
            agent = new
            BTServicAndDeviceAgent(UUID_STRING,UUID_SERVICE,this);

      }

      private void StartServer()
      {
            agent.StartServer();
            agent.SearchDevice();

      }

      private void StartClient()
      {
            agent.SearchDevice();
      }
```

```
	public static void main(String[] argv)

	{
		StartSystem startsystem = new StartSystem();
		startsystem.StartServer();

	}

	public void ReciveData(String message)
	{
		System.out.println(agent.Authorize.equals("authorize"));
		if(agent.Authorize.equals("authorize"))
			opendoor.Open();
	}

	public void FoundDevice(String[] devices)
	{

	}

}
```

Main Server

```
import java.io.*;
import java.net.*;

public class MainServer
{
private static long credit,credit2,BluetoothAddress;
private static int fare,stationId,Departure,Arrival;
private static String
balance1,bluetoothaddress,StationId,DepartureStation,ArrivalStation;
private DataBase database;

		public static void main(String args[]) throws Exception
		{
			System.out.println();
			Departure=0;
			Arrival=0;
			DataBase database = new DataBase();
			ServerSocket ss = new ServerSocket(12345);
			System.out.println("Server is operating");
			//    Socket s=ss.accept();
			while(true)
			{

				Socket s = ss.accept();
				BufferedReader br = new BufferedReader(
				new InputStreamReader (s.getInputStream()));
				bluetoothaddress=br.readLine();
				StationId=br.readLine();

	BluetoothAddress=Long.parseLong(bluetoothaddress);

			if(Departure==0)
			{
DepartureStation=database.station1;
credit=database.returnbalance(BluetoothAddress);
System.out.println("Customer ID "+BluetoothAddress);
```

```
System.out.println("Access system at station "+DepartureStation);
System.out.println("Credit in account "+credit+" baht");
                        if (credit>10)
                        {
            PrintStream op = new PrintStream(s.getOutputStream());
            op.println("You have "+credit+" baht in your account");
            op.println("authorize");
            Departure=stationId;
            System.out.println("Authorized to access system");
            System.out.println("----------------------------");
                                // authorize open door
                        }
                        else
                        {


            PrintStream op = new PrintStream(s.getOutputStream());
            op.println("Your credit is not enough to access the
            system.Please recharge your account.");
            op.println("nonauthorize");
            System.out.println("Not authorized to access system");
            System.out.println("--------------------------------");
                        }
stationId=Integer.parseInt(StationId);
Departure=stationId;
Arrival=0;
                    }
                    else
                    {
ArrivalStation=database.station2;
stationId=Integer.parseInt(StationId);
Arrival=stationId;
fare=database.returnfare(Departure,Arrival);
System.out.println("Customer ID "+BluetoothAddress);
System.out.println("Credit in account "+credit+" baht");
System.out.println("Enter at station "+DepartureStation);
System.out.println("Exit at station "+ArrivalStation);
System.out.println("Travel pass "+database.zone+" stops");

                        if(credit>fare)
                        {
            database.UpdateBalance();
            credit2=database.returnbalance(BluetoothAddress);
            PrintStream op = new PrintStream(s.getOutputStream());
            op.println("You have "+credit2+" baht in your account");
            op.println("authorize");
            System.out.println("Train fare "+"fare baht");
            System.out.println("Credit is enough to exit");
            System.out.println("Will have "+credit2+" baht left in
            account");
            System.out.println("Authorized to exit ");
            System.out.println("--------------------");
                         }
                         else
                         {
            PrintStream op = new PrintStream(s.getOutputStream());
            op.println("Your credit is not enough to exit this
            station.Contact the Office box");
            op.println("nonauthorize");
            System.out.println("Train fare "+fare+" baht");
            System.out.println("Credit is not enough to exit");
```

```
                System.out.println("Not authorized to exit");
                System.out.println("----------------------");
                            }
                      Departure=0;
                      }
            s.close();
            if (Thread.interrupted())
            break;
            }
            ss.close();
      }

}
```

DataBase

```
import java.sql.*;
import java.lang.*;
public class DataBase

{
      public String id,station1,station2,stringaddress;
      private int leftbalance, fare, balance, stationIndex1,
      stationIndex2,direction1,direction2;
      public int zone;
      private long bluetoothaddress;
      public int returnbalance(long bluetoothaddress)
      {
            this.bluetoothaddress=bluetoothaddress;
            stringaddress = bluetoothaddress+".0";
            try
            {
                  Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
                  Connection c =
            DriverManager.getConnection("Jdbc:Odbc:skytrain", "","");
                  Statement s = c.createStatement();
                  ResultSet r = s.executeQuery("SELECT * FROM
                  skytrain WHERE id="+bluetoothaddress);
                  while (r.next())
                  {
                        id=r.getString(1);
                        balance=r.getInt(2);
                  }
                  s.close();
                  c.close();
            }
            catch(Exception a)
            {
                  System.out.println(a);
            }
            return balance;
      }

      public int returnfare(int departure,int arrival)
      {
            try
            {
                  Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
                  Connection c =
      DriverManager.getConnection("Jdbc:Odbc:skytrain", "","");
```

```
                    Statement s = c.createStatement();
                    ResultSet r = s.executeQuery("SELECT * FROM station
WHERE Index="+departure);
                    while (r.next())
                    {
                          station1=r.getString(2);
                          stationIndex1=r.getInt(3);
                          direction1=r.getInt(4);
                    }
                    s.close();
                    c.close();
              }
               catch(Exception a)
              {
                    System.out.println(a);
              }
              try
              {
                    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
                          Connection c =
      DriverManager.getConnection("Jdbc:Odbc:skytrain", "","");
                    Statement s = c.createStatement();
                    ResultSet r = s.executeQuery("SELECT * FROM station
      WHERE Index="+arrival);
                    while (r.next())
                    {
                          station2=r.getString(2);
                          stationIndex2=r.getInt(3);
                          direction2=r.getInt(4);
                    }
                    if(direction1==direction2)
                          zone=Math.abs(stationIndex2-stationIndex1);
                    else
                          zone=stationIndex2+stationIndex1;
                    s.close();
                    c.close();
              }
               catch(Exception a)
              {
                    System.out.println(a);
              }

              try
              {
                    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
                    Connection c =
      DriverManager.getConnection("Jdbc:Odbc:skytrain", "","");
                    Statement s = c.createStatement();
                    ResultSet r = s.executeQuery("SELECT * FROM fare
WHERE distance="+zone);
                    while (r.next())
                    {
                          fare=r.getInt(2);
                    }
                    s.close();
                    c.close();
              }
              catch(Exception a)
              {
                    System.out.println(a);
              }
```

```
            return fare;
      }

      public void UpdateBalance() throws Exception
      {
            leftbalance=balance-fare;
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            Connection c =
DriverManager.getConnection("Jdbc:Odbc:skytrain", "", "");
            Statement s = c.createStatement();
            s.executeUpdate("UPDATE skytrain SET balance
="+leftbalance+" WHERE id="+bluetoothaddress);
            s.close(); c.close();
      }


}
```

Mobile

Skytrain2

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.io.*;
import javax.bluetooth.*;
import java.io.*;

public class Skytrain2 extends MIDlet implements BTCallback
,CommandListener{
      private static final String UUID_STRING =
"1111223344556677889 9AABBCCDDEEFF";
      private Image skytrain1,skytrain2,map;
      private ImageItem skytrainItem1,skytrainItem2,mapItem;
      private Display display;
      private Form mainUI;
      private ChoiceGroup devices = new ChoiceGroup(null,
Choice.EXCLUSIVE);
      private Command exitCommand = new Command("Exit", Command.EXIT,
2);
      private Command openCommand = new Command("Open",
Command.SCREEN, 1);
      private Command getCommand = new
Command("GetInfo",Command.SCREEN,1);
      private Command selectCommand = new Command("Ok",
Command.SCREEN, 1);
      private StringItem String1,String2;
      private BTServicAndDeviceAgent agent;
      public Skytrain2() {

            try{
                  skytrain1=Image.createImage("/skytrain1.png");
                  skytrain2=Image.createImage("/map.png");
                  map=Image.createImage("/skytrain2.png");
               }
              catch(java.io.IOException e){}
             skytrainItem1 = new
ImageItem("",skytrain1,ImageItem.LAYOUT_CENTER,null);
```

```
            skytrainItem2 = new
ImageItem("",skytrain2,ImageItem.LAYOUT_CENTER,null);
            mapItem=new
ImageItem("",map,ImageItem.LAYOUT_CENTER,null);
            display = Display.getDisplay(this);
            mainUI  = new Form("Skytrain");
            mainUI.setCommandListener(this);
            mainUI.addCommand(exitCommand);
            mainUI.addCommand(selectCommand);
    //      String2 = new StringItem("","Test2");
    }

      public void startApp()
      {
            mainUI.append(devices);
            mainUI.append(skytrainItem1);
            display.setCurrent(mainUI);
//          mainUI.append(String2);
            agent = new
BTServicAndDeviceAgent(UUID_STRING,"L2CAPchat",this);
            agent.StartServer();
            agent.SearchDevice();
      }
      public void pauseApp() {
      }
      public void destroyApp(boolean unconditionally)
      {
            agent.StopClient();
            agent.StopServer();
            agent.ReleaseResource();
      }
      public void ReciveData(String message)
      {
            String1=new StringItem("",message);
      }
      public void FoundDevice(String[] devicesname)
      {
            int aa = devicesname.length;
            for(int i=0;i<aa;i++)
            {
                  devices.append("Press 'Ok' to access ",null);
            }
            mainUI.append(devices);
            mainUI.addCommand(selectCommand);
      }
      public void commandAction(Command c, Displayable d) {
            if(c == exitCommand)
            {
                  agent.StopClient();
                  agent.StopServer();
                  agent.ReleaseResource();
                  notifyDestroyed();



            }
/*          else if(c== getCommand)
            {
                  mainUI.append(String1);
                  display.setCurrent(mainUI);
            }
```

```
*/

            else if (c == openCommand)
            {

//                  mainUI.append(String1);
                    agent.ToSendData("HI");
                    mainUI.append(String1);
                    display.setCurrent(mainUI);
            }
            else if (c == selectCommand)
            {   // use for select device from PC or other device
                    int index = devices.getSelectedIndex();
                    mainUI.deleteAll(); //delete choice group
                    mainUI.removeCommand(selectCommand);
                    mainUI.addCommand(exitCommand);
                    agent.SearchService(index);
                    mainUI.addCommand(openCommand);
      //            mainUI.append(String1);

            }
      }
}
```

BTServicAndDeviceDiscovery

```
import javax.bluetooth.*;
import java.util.*;
import java.io.*;
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.io.*;
public final class BTServicAndDeviceDiscovery implements
DiscoveryListener
{
      private Vector remoteDevices = new Vector();
      private ServiceRecord serviceRecord;
      private BTServicAndDeviceAgent callback;
      private String SERVICE_NAME;
      public BTServicAndDeviceDiscovery(String
SERVICE_NAME,BTServicAndDeviceAgent callback)
      {
            this.SERVICE_NAME = SERVICE_NAME;
            this.callback = callback;
      }
      public void servicesDiscovered(int transID, ServiceRecord[]
      servRecord)
      {
            for(int i = 0; i < servRecord.length; i++)
            {
            DataElement serviceNameElement =
servRecord[i].getAttributeValue(0x0100);//get the Service Name
            String serviceName =
            (String)serviceNameElement.getValue();
                  if(serviceName.equals(SERVICE_NAME))
                  {
                  serviceRecord = servRecord[0];
                  }
            }
```

```
    }

    public void serviceSearchCompleted(int transID, int respCode)
    {
        String message = null;
        if (respCode ==
        DiscoveryListener.SERVICE_SEARCH_DEVICE_NOT_REACHABLE) {
        message = "Device not reachable";
        }
        else if (respCode ==
        DiscoveryListener.SERVICE_SEARCH_NO_RECORDS) {
        message = "Service not available";
        }
        else if (respCode ==
        DiscoveryListener.SERVICE_SEARCH_COMPLETED)
        {
            message = "Service search completed";
        }
        else if (respCode ==
        DiscoveryListener.SERVICE_SEARCH_TERMINATED) {
        message = "Service search terminated";
        }
        else if (respCode ==
        DiscoveryListener.SERVICE_SEARCH_ERROR) {
            message = "Service search error";
        }
        callback.serviceSearchFinished(serviceRecord, message);
    }

    public void inquiryCompleted(int discType)
    {
        String message = null;
        if (discType == INQUIRY_COMPLETED)
        {
                message = "Inquiry completed";
        }
        else if (discType == INQUIRY_TERMINATED)
        {
                message = "Inquiry terminated";
        }
        else if (discType == INQUIRY_ERROR)
        {
                message = "Inquiry error";
        }

        RemoteDevice[] devices = new
        RemoteDevice[remoteDevices.size()];
        for(int i = 0; i < remoteDevices.size(); i++)
        {
            devices[i] =
            (RemoteDevice)remoteDevices.elementAt(i);
        }
        callback.deviceInquiryFinished(devices, message);
    }

public void deviceDiscovered(RemoteDevice btDevice, DeviceClass cod)
    {
        remoteDevices.addElement(btDevice);
    }
}
```

BTServicAndDeviceAgent

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.io.*;
import javax.bluetooth.*;
import java.io.*;

public final class BTServicAndDeviceAgent
{
      private String UUID_STRING;
      private String SERVICE_NAME;
      private LocalDevice localDevice;
      private DiscoveryAgent agent;
      private L2CAPConnection conn;
      private RemoteDevice[] remoteDevices;
      private boolean running = false;
      private BTCallback callback;
      private String msg;
      private BTServicAndDeviceDiscovery discoverer;
      private String[] devices;
      private BTServicAndDeviceAgent(){ }

      public BTServicAndDeviceAgent(String UUID_STRING,String
SERVICE_NAME,BTCallback callback)
      {
            this.UUID_STRING = UUID_STRING;
            this.SERVICE_NAME = SERVICE_NAME;
            this.callback = callback;
discoverer = new BTServicAndDeviceDiscovery(SERVICE_NAME,this);
            try
            {
                  localDevice = LocalDevice.getLocalDevice();
                  agent = localDevice.getDiscoveryAgent();
            }
            catch(BluetoothStateException bse)
            {
                  bse.printStackTrace();
            }
      }

      public void StartServer()
      {
            new Thread()
            {
                  public void run()
                  {
                        try
                        {
                  localDevice.setDiscoverable(DiscoveryAgent.GIAC);
                  L2CAPConnectionNotifier notifier =
                  (L2CAPConnectionNotifier)Connector.open("btl2cap://
                  localhost:" + UUID_STRING + ";name=L2CAPchat");
                  ServiceRecord record =
                  localDevice.getRecord(notifier);
String connURL =
record.getConnectionURL(ServiceRecord.NOAUTHENTICATE_NOENCRYPT,
false);
                  conn = notifier.acceptAndOpen();//record is saved
```

```
                new Thread()
                {
                public void run()
                {
                        running = true;
                        ToReciveData();
                }
                }.start();
                }
                catch(IOException ioe)
                {
                        ioe.printStackTrace();
                }
        }
        }.start();
    }

    public void StopServer()
    {
        running = false;
    }
    public void SearchDevice()
    {
        try
        {
                    //non-blocking

    agent.startInquiry(DiscoveryAgent.GIAC, discoverer);
        }
        catch(BluetoothStateException bse)
        {
            bse.printStackTrace();
        }
    }
    public void SearchService(int index)
    {
        int[] attrSet = {0x0100};//return service name attribute
        UUID[] uuidSet = new UUID[1];
        uuidSet[0] = new UUID(UUID_STRING, false);
        try
        {
            agent.searchServices(attrSet, uuidSet,
            remoteDevices[index], discoverer);
        }
        catch(BluetoothStateException bse)
        {
            bse.printStackTrace();
        }
    }
    public void StopClient()
    {
        running = false;
    }

    public void deviceInquiryFinished(RemoteDevice[] remoteDevices,
String message)
    {
        this.remoteDevices = remoteDevices;
        devices = new String[remoteDevices.length];
        for(int i = 0; i < remoteDevices.length; i++)
        {
```

```
                    try
                    {
                String name = remoteDevices[i].getFriendlyName(false);
                        devices[i] = name;
                    }
                    catch(IOException ioe)
                    {
                        ioe.printStackTrace();
                    }
                }
                callback.FoundDevice(devices);
        }

        public void serviceSearchFinished(ServiceRecord serviceRecord,
String message)
        {
                String url =
serviceRecord.getConnectionURL(ServiceRecord.NOAUTHENTICATE_NOENCRYPT
, false);
                    try
                    {
                        conn = (L2CAPConnection)Connector.open(url);
                        new Thread()
                            {
                                public void run(){
                                running = true;
                                ToReciveData();
                                }
                            }.start();
                    }
                    catch(IOException ioe)
                    {
                        ioe.printStackTrace();
                    }
        }

        public void ToReciveData()
        {
                while(running)
                {
                    try
                    {
                        if(conn.ready())
                        {
                        int receiveMTU = conn.getReceiveMTU();
                        byte[] data = new byte[receiveMTU];
                        int length = conn.receive(data);
                        String message = new String(data, 0, length);
                        callback.ReciveData(message);
                        }
                    }
                    catch(IOException ioe)
                    {
                        ioe.printStackTrace();
                    }
                }
        }

        public void ToSendData(String message)
        {
                msg = null;
```

```
                msg = message;
                new Thread()
                {
                        public void run()
                        {
                             try
                             {
                               byte[] data = msg.getBytes();
                               int transmitMTU = conn.getTransmitMTU();
                               if(data.length <= transmitMTU)
                               {
                                     conn.send(data);
                               }
                               else
                               {
                               System.out.println("Message too long!");
                               }
                             }
                             catch(IOException ioe)
                             {
                               ioe.printStackTrace();
                             }
                        }
                }.start();
}

        public void ReleaseResource()
        {
                try
                {
                        if(localDevice != null) localDevice = null;
                        if(agent != null) agent = null;
                        if(remoteDevices != null) remoteDevices = null;
                        if(conn != null) conn.close();
                }
                catch(IOException ioe)
                {
                        ioe.printStackTrace();
                }

        }
}
```

BTCallback

```
public interface BTCallback
{
        public void ReciveData(String message);
        public void FoundDevice(String[] devicesname);
}
```