บทที่ ๑๐ การรับเข้าและส่งออกข้อมูล

ตัวอย่างโปรแกรมที่ผ่านมามักมีการกำหนดข้อมูลที่ต้องการใช้ใส่ไว้ในโปรแกรมอย่างตายตัว โปรแกรมที่ใช้งานจริงนั้นส่วนใหญ่แล้วมักมีความจำเป็นต้องนำข้อมูลจากภายนอกเข้าสู่โปรแกรมที่กำลังทำงาน เมื่อรับข้อมูลเข้ามาแล้วมักมีการประมวลผลข้อมูลนั้นแล้วแสดงผลลัพธ์ออกมาทางใดทางหนึ่งหรือเก็บข้อมูลไว้เพื่อใช้งานต่อ ข้อมูลที่ใช้ติดต่อกับมนุษย์จะอยู่ในรูปแบบที่มนุษย์เข้าใจได้ ส่วนข้อมูลที่จัดเก็บไว้เพื่อนำไปประมวลผลต่อไปไม่จำเป็นต้องจัดเก็บในรูปแบบที่มนุษย์เข้าใจได้ง่ายแต่มักจัดเก็บในรูปแบบที่เครื่องทำงานได้อย่างมีประสิทธิภาพ

ข้อมูลบางส่วนอาจกำหนดค่าเอาไว้ในโปรแกรมโดยผู้เขียนโปรแกรมตั้งแต่แรก ข้อมูลบางส่วนไม่สามารถกำหนดไว้ตั้งแต่แรกเริ่มได้ จำเป็นต้องใช้ข้อมูลใหม่ซึ่งเปลี่ยนไปตามสถานการณ์ ตัวอย่างเช่น โปรแกรมสำหรับแปลงค่าความยาวจากหน่วยที่เป็นนิ้วให้เป็นเซ็นติเมตร โปรแกรมอาจถามความยาวหน่วยเป็นนิ้วจากผู้ใช้โปรแกรม ตัวเลขความยาวที่ผู้ใช้กรอกเข้ามานั้น โปรแกรมไม่สามารถล่วงรู้ล่วงหน้าได้ว่าจะเป็นเท่าใด จำเป็นต้องถามสดขณะที่โปรแกรมกำลังทำงาน เมื่อผู้ใช้ป้อนตัวเลขเข้ามา โปรแกรมจะรับค่าตัวเลขนี้ไปประมวลผลโดยคูณด้วย 2.54 แล้วแสดงผลลัพธ์ที่คำนวณได้ออกมาให้ผู้ใช้เห็น ค่า 2.54 เป็นค่าคงที่ที่เรากำหนดตายตัวลงไปในโปรแกรม สำหรับข้อมูลที่ผู้ใช้ป้อนเข้ามานั้น เราต้องหาวิธีการใดวิธีการหนึ่งที่จะรับข้อมูลนั้นเข้าสู่โปรแกรม ส่วนการแสดงผลลัพธ์นั้น เราพบเห็นตัวอย่างมามากพอสมควรแล้ว ในบทนี้จะกล่าวถึงการรับข้อมูลเข้า และส่งข้อมูลออกด้วยวิธีการต่างๆ

การรับข้อมูลเข้าทางสายคำสั่ง(Command Line Input)

จาวามีกลไกสำหรับรับข้อมูลจากผู้ใช้โปรแกรม ในขณะที่สั่งรันโปรแกรม โดยการใส่ข้อมูลตามหลังชื่อคลาสที่ต้องการรันผู้ใช้โปรแกรม ตัวอย่างเช่นเราต้องการรันโปรแกรม Sum ซึ่งทำหน้าที่บวกเลข เราสามารถส่งข้อมูลให้โปรแกรมนี้ขณะเรียกรันโปรแกรมได้ดังตัวอย่างต่อไปนี้

```
java  Sum 11  12  13  18.9  34  23.5
```

ตัวอย่างนี้เป็นบรรทัดคำสั่ง (command line) ที่ใช้ในการรันโปรแกรม Sum ข้อมูลที่เขียนตามหลังชื่อคลาสจะถูกแปลงให้อยู่ในรูปของแถวลำดับของสายอักขระ ที่อยู่ของแถวลำดับนี้จะถูกส่งให้โปรแกรมทางพารามิเตอร์ของเมธอด main

```
๑. class Sum {
๒.     public static void main(String args[ ]) {
๓.         if ( args.length == 0) {
๔.             System.out.println( "Usage: java Sum value1 ... valueN");
๕.             System.out.println( "value must be number");
๖.             System.exit(1);
๗.         }
๘.         else {
๙.             double total=0;
```

```
๑๐.             try {
๑๑.                  for (int i=0; i < args.length ; i++ )
๑๒.                  total+=Double.parseDouble(args[i]);
๑๓.             }
๑๔.             catch (NumberFormatException e) {
๑๕.                  System.err.println(
๑๖.                            "command line arguments must be numbers");
๑๗.                  System.exit(2);
๑๘.             }
๑๙.              System.out.print("Sum of all values entered is " + total);
๒๐.         }
๒๑.     }
๒๒. }
```

ตัวอย่างโปรแกรมนี้ รับข้อมูลผ่านทางพารามิเตอร์ของเมธอด `main` เมื่อเมธอด `main` เริ่มทำงานจะจะตรวจสอบว่ามีการส่งข้อมูลผ่านทางพารามิเตอร์เข้ามาหรือไม่ โดยการดูจำนวนจากตัวแปร `args.length` ซึ่งถ้ามีค่าเป็นศูนย์แสดงว่าไม่มีการส่งข้อมูลใดๆมาให้ จะแสดงวิธีใช้โปรแกรมและยุติการทำงานของโปรแกรม แต่ถ้าค่าของ `args.length` มีค่ามากกว่าศูนย์แสดงว่ามีการรับข้อมูลผ่านทางบรรทัดคำสั่งเข้ามา จึงทำการประมวลผลต่อไป ซึ่งโปรแกรมนี้จะแปลงข้อมูลซึ่งรับเข้ามาในรูปของสายอักขระให้เป็นชนิด `double` แล้วนำไปรวมยอดในตัวแปร `total` โปรแกรมจะทำซ้ำจนกระทั่งครบข้อมูลที่รับเข้ามาแล้วจึงแสดงผลค่าผลรวม

การแปลงจากสายอักขระเป็นข้อมูลชนิด `double` นั้นใช้เมธอด `Double.parseDouble( )` ซึ่งถ้าข้อมูลที่ต้องการแปลงไม่ใช่ตัวเลขจะเกิดสิ่งผิดปรกติ `NumberFormatException` ขึ้น โปรแกรมจะหยุดทำงานไปเอง แต่เราต้องการจัดการกับสิ่งผิดปรกตินั้นเองจึงมีการดักจับสิ่งผิดปรกติชนิดนี้เพื่อแจ้งข้อความผิดพลาดให้ผู้ใช้โปรแกรมทราบ

คลาส File

คลาส File เป็นคลาสสำหรับสร้างอ๊อบเจ็กท์ซึ่งจะทำการห่อหุ้มชื่อแฟ้มหรือชื่อพาธเอาไว้ อ๊อบเจ็กท์ที่ได้สามารถส่งเป็นพารามิเตอร์ไปให้ตัวสร้างของคลาสอื่นเพื่อบ่งบอกแฟ้มที่เราต้องการใช้ คลาสนี้ไม่เกี่ยวกับการนำเข้าหรือส่งออกข้อมูลโดยตรง แต่ใช้สำหรับสอบถามข้อมูลเกี่ยวกับแฟ้มหรือ directory คลาสนี้มีเมธอดหลายเมธอดสำหรับสอบถามข้อมูลต่างๆ เกี่ยวกับแฟ้ม เช่นถามวันที่ที่มีการปรับปรุงแก้ไขล่าสุด ถามขนาดของแฟ้มเป็นต้น และมีเมธอดจำนวนหนึ่งสำหรับใช้ในการจัดการแฟ้มเช่น การลบแฟ้ม การเปลี่ยนชื่อแฟ้มเป็นต้น

กระแสข้อมูล (Stream)

ข้อมูลที่รับเข้าหรือส่งออกจะไหลอย่างต่อเนื่องไปตามทางของมันตามลำดับเปรียบเสมือนสายธาร จึงมีการนำคำว่า "กระแส" (stream) มาใช้กับการรับเข้าหรือส่งออกข้อมูลในลักษณะนี้

คลาสสำหรับนำเข้าและส่งออกข้อมูล

ในชุด JDK มีคลาสและเมธอดที่ช่วยในการรับเข้าและส่งออกข้อมูลอยู่เป็นจำนวนมากให้เลือกใช้ ข้อมูลอาจอยู่ในรูปของข้อความซึ่งประกอบด้วยรหัสอักขระตามมาตรฐาน ASCII หรืออยู่ในรูปของเลขฐานสองตามประเภทของข้อมูล

ทางรับเข้าและส่งออกมาตรฐาน

ในระบบ UNIX หรือ MS-DOS จะมีการเตรียมช่องทางสำหรับรับข้อมูลเข้าและส่งข้อมูลออกในรูปของข้อความไว้จำนวนหนึ่งเป็นมาตรฐาน ให้ผู้ใช้และนักเขียนโปรแกรมใช้งานได้อย่างสะดวก โดยทั่วไปแล้ว ในการรับเข้าและส่งออกข้อมูลนั้น จำเป็นต้องทำการเปิด (open) ช่องทางเอาไว้ก่อนจึงจะรับเข้าข้อมูลได้ และเมื่อใช้งานเสร็จแล้วต้องทำการปิด (close) ช่องทางนั้น การเปิดเป็นการเตรียมความพร้อมและตระเตรียมทรัพยากรต่างๆที่จำเป็นเช่นการตรวจสถานะของอุปกรณ์ การจัดสรรหน่วยความจำสำหรับเก็บพักข้อมูล เมื่อไม่มีความจำเป็นต้องใช้ช่องทางรับเข้านั้นอีกต่อไป เราควรทำการปิดเพื่อคืนทรัพยากรต่างๆ ที่ใช้กลับคืนให้กับระบบ เพื่ออำนวยความสะดวกให้กับผู้ใช้และนักเขียนโปรแกรม มีช่องทางที่ระบบเปิดรอไว้แล้วเพื่อให้ใช้ได้ทันทีโดยที่นักเขียนโปรแกรมไม่ต้องทำการเปิดหรือปิดด้วยตนเองให้ยุ่งยาก ช่องทางนี้เรียกว่า *ทางรับเข้าหรือส่งออกมาตรฐาน* (standard Input/Output หรือเรียกสั้นสั้นว่า standard IO)

ทางรับเข้าหรือส่งออกมาตรฐานมี ๓ กระแสให้ใช้ดังนี้

๑. กระแสรับเข้ามาตรฐาน (standard input stream) เป็นช่องทางสำหรับรับข้อมูลเข้าแบบปรกติ ซึ่งโดยทั่วไปแล้วจะรับเข้าทางแผงแป้นอักขระ(keyboard)

๒. กระแสส่งออกมาตรฐาน (standard output stream) เป็นช่องทางสำหรับส่งข้อมูลออกแบบปรกติ โดยทั่วไปแล้วจะออกทางจอภาพ

๓. กระแสข้อผิดพลาดมาตรฐาน (standard error stream) เป็นช่องทางสำหรับส่งข้อความบ่งบอกข้อผิดพลาดไปแสดงผล โดยปรกติแล้วจะแสดงออกทางจอภาพ

คุณสมบัติที่สำคัญของทางรับเข้าหรือส่งออกมาตฐานคือ ช่องทางเหล่านี้เปิดโดยอัตโนมัติ นักเขียนโปรแกรมไม่จำเป็นต้องเปิดก่อนใช้ และช่องทางเหล่านี้สามารถบังคับให้เปลี่ยนทิศทาง (redirect) จากภายนอกโปรแกรมได้โดยที่โปรแกรมไม่จำเป็นต้องรู้และโปรแกรมไม่จำเป็นต้องจัดการเอง เช่นถ้าเราเขียนโปรแกรมสำหรับรับข้อมูลเข้าทางกระแสรับเข้ามาตรฐาน โดยปรกติแล้วเราสามารถป้อนข้อมูลเข้าสู่โปรแกรมทางแผงแป้นอักขระ แต่ถ้าหากมีการบังคับเปลี่ยนทิศทางของกระแสรับเข้ามาตรฐานแล้ว ข้อมูลเข้าอาจมาจากทางอื่นที่ไม่ใช่แผงแป้นอักขระโดยตรงก็ได้ เช่นมาจากแฟ้มข้อมูล หรือรับมาจากอุปกรณ์สื่อสาร หรืออาจรับมาจากโปรแกรมอื่นอีกทอดก็ได้ กระแสส่งออกมาตรฐานก็เช่นเดียวกัน สามารถถูกบังคับเปลี่ยนทิศทางไปออกที่อื่นนอกเหนือจากจอภาพได้ เช่นบังคับให้ไปออกที่เครื่องพิมพ์ หรือ ส่งลงแฟ้มข้อมูล ส่งไปยังอุปกรณ์สื่อสาร หรืออาจส่งไปให้โปรแกรมอื่นรับช่วงไปประมวลผลต่อก็ได้ ส่วนกระแสข้อผิดพลาดมาตรฐานนั้นไม่สามารถบังคับเปลี่ยนทิศทางได้ง่ายเหมือน ๒

กระแสแรกที่กล่าวไปแล้ว ทั้งนี้ก็ด้วยเหตุผลเพื่อการแยกแยะข้อมูลระหว่างข้อมูลออกปรกติกับข้อผิดพลาดนั่นเอง ถ้าเราบังคับเปลี่ยนทิศของข้อมูลส่งออกให้ไปพิมพ์ที่เครื่องพิมพ์หรือลงแฟ้มข้อมูล ข้อความแสดงข้อผิดพลาดที่ส่งออกทางกระแสข้อผิดพลาดมาตรฐานจะไม่ไปด้วย

การรับข้อความเข้าทางกระแสรับเข้ามาตรฐาน

ในคลาส `System` มีตัวแปรอ้างอิง `in` ใช้สำหรับอ้างอิงไปถึงอ๊อบเจ็กต์ของคลาส `InputStream` ซึ่งเป็นคลาสทั่วไปสำหรับนำข้อมูลเข้า ในกรณีของกระแสรับเข้ามาตรฐานจะมีการสร้างอ๊อบเจ็กต์ของคลาสนี้รอไว้แล้วเพื่อเตรียมพร้อมให้ใช้กระแสรับเข้ามาตรฐานได้ทันที โดยนักเขียนโปรแกรมไม่จำเป็นต้องสร้างอ๊อบเจ็กต์ขึ้นเอง ที่อยู่ของอ๊อบเจ็กต์ที่สร้างขึ้นจะนำมาเก็บไว้ในตัวแปร `in` ให้เราใช้งานได้อย่างสะดวก

ในคลาส `System` มีการประกาศตัวแปร `in` ไว้ดังนี้

```
public static final InputStream in
```

เนื่องจาก `in` เป็นตัวแปรที่อ้างอิงไปยังคลาส `InputStream` ดังนั้นเราจึงสามารถใช้เมธอดต่างๆ ที่เขียนไว้ในคลาส `InputStream` และคลาสบนของมันได้ คลาสนี้มีหลายเมธอดให้ใช้ แต่ในที่นี้จะนำเมธอด `read` มาใช้เพียงเมธอดเดียว เมธอด `read` ทำหน้าที่ในการอ่านอักขระเข้าทางกระแสรับเข้าทีละตัวหรือทีละไบต์ เมธอดนี้มีการประกาศไว้ดังนี้

```
public abstract int read() throws IOException
```

เมธอดนี้ส่งค่ากลับคืนมาเป็นประเภท `int` มีค่าอยู่ระหว่าง 0 - 255 ซึ่งเป็นค่าของอักขระต่างๆ ที่กำหนดไว้ในรหัสมาตรฐานของสหรัฐอเมริกาเพื่อการสับเปลี่ยนสารสนเทศ (ASCII) หากไม่มีข้อมูลให้อ่านเนื่องจากถึงจุดจบของกระแสแล้วจะคืนค่าเป็น -1 อักขระที่รับเข้ามาจะถูกนำไปเก็บพักไว้ในหน่วยความจำ จะยังไม่ส่งคืนค่าไปให้ผู้เรียกเมธอดนี้จนกว่าจะพบรหัสควบคุมที่บอกว่าขึ้นบรรทัดใหม่ (LF : line feed มีค่าเป็น 10 ในเลขฐานสิบตามมาตรฐาน ASCII) วัตถุประสงค์หนึ่งของการเก็บพักข้อมูลเอาไว้ก่อนยังไม่คืนกลับไปยังผู้เรียกใช้เมธอดในทันทีนั้นก็เพื่อเปิดโอกาสให้ผู้ใช้โปรแกรมสามารถแก้ไขข้อมูลที่เคาะเข้าทางแผงแป้นอักขระได้ หากยังไม่เคาะ <Enter> ผู้ใช้สามารถกดปุ่ม <Backspace> ถอยกลับไปแก้ไขข้อมูลที่เคาะเข้าไปก่อนหน้านี้ได้

โปรดสังเกตุว่า ถึงแม้ข้อมูลที่อยู่ในกระแสจะเป็นไบต์ซึ่งมีค่าไม่เกิน 255 ก็ตาม ค่าที่ส่งกลับคืนมาจะได้รับการส่งเสริม(promote) ให้เป็น `int` นอกจากนี้ยังมีการประกาศบังคับเอาไว้ว่าผู้ที่จะนำเมธอดนี้ไปใช้จะต้องมีการดักจับสิ่งผิดปรกติชนิด `IOException` ด้วย

ตัวอย่างที่ ๑๕-๑ แสดงการใช้เมธอด `read` สำหรับอ่านข้อมูลจากกระแสรับเข้ามาตรฐาน โปรแกรมนี้จะทำซ้ำเป็นวงวน มีการรับข้อมูลเข้าแล้วนำไปแสดงผล ทำวนเช่นนี้จนกว่าโปรแกรมจะถูกบังคับให้หยุดทำงาน บรรทัดที่ ๘ มีการเรียกใช้เมธอด `read` ผ่านทางตัวแปร `in` ซึ่งเป็นตัวแปรสถิตอยู่ในคลาส `System` เมธอดนี้ส่งค่ากลับคืนเป็น `int` นำไปเก็บไว้ในตัวแปร `c` ซึ่งมีการประกาศไว้ในบรรทัดที่ ๕ มีประเภทเป็น `int` หลังจากอ่านข้อมูลเข้ามาแล้วจะตรวจสอบว่าข้อมูลที่อ่านได้มีค่าเป็น `-1` ซึ่งหมายถึงไม่มีข้อมูลให้อ่านอีกต่อไปแล้ว โปรแกรม

ก็จะออกจากวงวนโดยข้อความสั่ง break ในบรรทัดที่ ๑๐ ส่วนบรรทัดที่ ๑๑ ใช้สำหรับแสดงค่าข้อมูลที่รับเข้าไปโดยแสดงออกมาตามประเภทของข้อมูลคือ `int` ซึ่งเป็นค่าตัวเลขที่กำหนดให้กับตัวอักขระต่างๆ ตามรหัสมาตรฐาน ASCII เช่นถ้าเราป้อนข้อมูลเป็นตัว A เข้าไปจะแสดงเป็นค่าตัวเลข 65 และถ้าป้อนข้อมูลเป็นตัวอักขระ 0 (ศูนย์) เข้าไปจะแสดงเป็นค่าตัวเลข 48 เมื่อเราเคาะอักขระที่ไม่ใช่ตัวควบคุม ข้อมูลของอักขระนั้นจะถูกนำไปเก็บสะสมไว้ในที่พัก จนกว่าจะพบตัวควบคุมขึ้นบรรทัดใหม่ซึ่งถ้าเราป้อนเข้าทางแผงแป้นอักขระคือตัว <Enter> นั่นเอง ข้อมูลที่เก็บสะสมไว้ในที่พักจะถูกทะยอยส่งเป็นค่าคืนไปให้ผู้เรียกเมธอด ค่าที่ส่งไปให้จะรวมรหัสควบคุมที่ได้จากการเคาะแป้น <Enter> ด้วย ทุกครั้งที่เคาะแป้น <Enter> จะได้รหัส ASCII ที่เป็นรหัสควบคุม ๒ ตัวคือ CR (Carriage Return) และ LF ซึ่งมีค่าเป็น 13 และ 10 ในเลขฐานสิบตามลำดับ

บรรทัดที่ ๑๔ จะทำงานก็ต่อเมื่อเกิดสิ่งผิดปรกติชนิด `IOException` ขึ้นในบล็อก `try` บรรทัดนี้จะส่งข้อมูลออกทางกระแสข้อผิดพลาดมาตรฐาน โดยใช้เมธอด `println` ของอ๊อบเจ็กต์ซึ่งอ้างอิงโดยตัวแปร `err` ที่กำหนดไว้เป็นตัวแปรสถิตในคลาส `System` จะเห็นว่ามีวิธีใช้ทำนองเดียวกับ `System.out.println` ต่างกันแต่เพียงช่องทางออกเท่านั้น `System.`**`out`**`.println` ใช้สำหรับส่งข้อมูลออกทางกระแสส่งออกมาตรฐาน ส่วน `System.`**`err`**`.println` ส่งออกทางกระแสข้อผิดพลาดมาตรฐาน

ตัวอย่างที่ ๑๕-๑ แสดงการใช้เมธอด `read` สำหรับอ่านข้อมูลจากกระแสรับเข้ามาตรฐาน

```
๒๓. import java.io.IOException;
๒๔. class Input1 {
๒๕.     public static void main ( String args [] ) {
๒๖.         try {
๒๗.             int c;
๒๘.             System.out.println("Enter character(s) and press Enter; to
    terminate program press Ctrl-Z");
๒๙.             while ( true ) {
๓๐.                 c = System.in.read();
๓๑.                 if (c == -1)
๓๒.                     break;
๓๓.                 System.out.println(c);
๓๔.             }
๓๕.         } catch (IOException e) {
๓๖.             System.err.println( e );
๓๗.         }
๓๘.     }
๓๙. }
```

ภาพผลลัพธ์ที่ ๑๕-๑ แสดงตัวอย่างผลลัพธ์จากการทำงานของตัวอย่างโปรแกรมที่ ๑๕-๑

```
Enter character(s) and press Enter; to terminate program press Ctrl-Z
A
65
13
10
0
```

```
48
13
10
Thai
84
104
97
105
13
10
```

ภาพผลลัพธ์ที่ ๑๕-๑ แสดงตัวอย่างผลลัพธ์จากการทำงานของตัวอย่างโปรแกรมที่ ๑๕-๑ เมื่อมีการป้อนข้อมูลตัวแรกเป็น A แล้วเคาะ <Enter> จะแสดงค่าของตัว A ตามมาตรฐาน ASCII เป็น 65 แล้วแสดงผลตามด้วย 13 และ 10 ซึ่งเป็นผลมาจากการเคาะ <Enter> หลังจากนั้นเคาะเลข 0 (ศูนย์) แล้วตามด้วย <Enter> จะได้ผลลัพธ์เป็น 48 13 และ 10 ตามลำดับ ต่อจากนั้นพิมพ์คำว่า Thai แล้วเคาะ <Enter> จะได้ผลลัพธ์เป็น 84 ซึ่งเป็นค่าของตัว T ตามด้วย 104 ค่าของตัว h 97 ค่าของตัว a 105 ค่าของตัว i แล้วจบด้วย 13 และ 10 ค่าของ <Enter> ตามลำดับ

เนื่องจากโปรแกรมนี้อ่านข้อมูลเข้าทางกระแสรับเข้ามาตรฐาน ดังนั้นเราสามารถส่งข้อมูลเข้าทางอื่นที่ไม่ใช้แผงแป้นอักขระได้ทันทีโดยไม่ต้องแก้ไขโปรแกรม เช่นสมมติว่าเรามีแฟ้มข้อความอยู่แล้วชื่อว่า test.txt เราสามารถให้โปรแกรมนี้อ่านข้อมูลจากแฟ้มนี้ได้โดยการสั่งเปลี่ยนทิศทางของข้อมูลเข้าให้อ่านจากแฟ้มแทนที่จะเป็นแผงแป้นอักขระขณะที่เรียกโปรแกรมที่ DOS prompt ได้ดังนี้

```
java  Input1  <  test.txt
```

เครื่องหมายน้อยกว่า (<) ใช้สำหรับบังคับเปลี่ยนทิศทางบอกให้นำข้อมูลจากแฟ้ม test.txt เข้ามาเป็นอินพุตแทนแผงแป้นพิมพ์

โปรแกรมนี้ทำงานเป็นวงวน อ่านข้อมูลแล้วแสดงผล ถ้าอ่านข้อมูลมาจากแฟ้ม เมื่ออ่านพบรหัสฐานสิบเป็นค่า 26 ซึ่งเป็นรหัสควบคุมบอกจุดจบของแฟ้มประเภทข้อความ หรือเมื่ออ่านไปจนจบแฟ้มไม่มีข้อมูลให้อ่านอีกต่อไป เมธอด read จะส่งค่ากลับคืนเป็น -1 ทำให้โปรแกรมออกจากวงวนและจบโปรแกรมไปในที่สุด ในกรณีที่อ่านข้อมูลเข้าทางแผงแป้นอักขระ โปรแกรมจะหยุดรอรับข้อมูลจนกว่าเราจะป้อนเข้าไปเมื่อเราเคาะ <Enter> โปรแกรมจึงทำงานต่อไปและย้อนกลับมาหยุดรอรับข้อมูลทางแผงแป้นอัขระอีก วนซ้ำอย่างนี้ไปเรื่อยๆ จนกว่าเราจะป้อนรหัสควบคุมบ่งบอกจุดจบของข้อมูลซึ่งทำได้โดยป้อนตัวควบคุม Ctrl-Z (กดปุ่ม Ctrl ค้างไว้แล้วกดปุ่มตัว Z) ซึ่งให้ค่าเป็นรหัสฐานสิบเป็น 26 โปรแกรมจะยุติการทำงานอย่างปรกติ หรือเราอาจบังคับโปรแกรมให้ยุติการทำงานลงกลางคันโดยป้อน Ctrl-C ก็ได้ วิธีนี้ใช้ได้ผลถ้าโปรแกรมมีการรับข้อมูลทางแผงแป้นอักขระโดยกระแสรับเข้ามาตรฐาน ซึ่งมีการตรวจสอบตัว Ctrl-C อยู่ด้วย ถ้าพบตัวนี้โปรแกรมจะยุติการทำงานทันทีโดยไม่คืนกลับไปยังผู้เรียกเมธอด

ตัวอย่างที่ ๑๕-๑ แสดงผลลัพธ์เป็นค่าตัวเลขของอักขระที่ป้อนเข้าไปและพิมพ์ค่าของ <Enter> ออกมาด้วย ทำให้ดูผลลัพธ์ยากว่าที่เราป้อนเข้าไปนั้นแสดงผลออกมาเป็นค่าใดกันแน่ถ้าเราป้อนข้อมูลเข้าไปหลายตัว จึงปรับปรุงโปรแกรมให้แสดงผลให้ดูง่ายขึ้นได้เป็นตัวอย่างที่ ๑๕-๒

ตัวอย่างที่ ๑๕-๒ ปรับปรุงมาจากตัวอย่างที่ ๑๕-๑ เพื่อให้แสดงผลลัพธ์เป็นตัวอักษรออกมาด้วย

```
๑. import java.io.IOException;
๒. class Input2 {
๓.     public static void main ( String args [] ) {
๔.         try {
๕.             int c;
๖.             System.out.println("Enter character(s) and press Enter; to
   terminate program press Ctrl-Z");
๗.             while ( true ) {
๘.                 c = System.in.read();
๙.                 if (c == -1)
๑๐.                    break;
๑๑.                if ( c == 13 ) {
๑๒.                    continue;
๑๓.                }
๑๔.                if ( c == 10 ) {
๑๕.                    System.out.println();
๑๖.                    continue;
๑๗.                }
๑๘.                System.out.print(c);
๑๙.                System.out.print("=");
๒๐.                System.out.print((char)c);
๒๑.                System.out.print(" ");
๒๒.            }
๒๓.        } catch (IOException e) {
๒๔.            System.err.println( e );
๒๕.        }
๒๖.    }
๒๗.}
```

ตัวอย่างที่ ๑๕-๒ ปรับปรุงจากตัวอย่างที่ ๑๕-๑ เล็กน้อยคือถ้าค่าที่คืนมาจากเมธอด `read` เป็น 13 (CR) จะไม่แสดงผล ถ้าค่าที่คืนมาเป็น 10 (LF) จะสั่งขึ้นบรรทัดใหม่ ถ้าเป็นตัวอื่นนอกจาก ๒ ตัวนี้จะแสดงค่ารหัสเลขฐานสิบและค่าอักขระของมันออกมา ในบรรทัดที่ ๒๐ นำค่าของ c มาหล่อหลอมให้เป็นประเภท `char` เพื่อให้พิมพ์เป็นอักขระแทนที่จะพิมพ์เป็นค่าในเลขฐานสิบ หลังจากพิมพ์ค่าและตัวอักขระเหล่านี้แล้วจะยังไม่ขึ้นบรรทัดใหม่ในทันที จะขึ้นบรรทัดใหม่ก็ต่อเมื่อพบค่า 10 ตามที่มีการตรวจสอบและพิมพ์บรรทัดใหม่ในบรรทัด ๑๔ และ ๑๕ ตามลำดับ ทั้งนี้เพื่อให้สัมพันธ์กับการเคาะ <Enter> และแสดงผลอย่างกระชับ สวยงาม

ภาพผลลัพธ์ที่ ๑๕-๒ แสดงตัวอย่างผลลัพธ์จากการทำงานของตัวอย่างที่ ๑๕-๒

```
Enter character(s) and press Enter; to terminate program press Ctrl-Z
A
65=A
0
48=0
Thai
```

```
84=T 104=h 97=a 105=i
```

ภาพผลลัพธ์ที่ ๑๕-๒ เป็นผลลัพธ์ที่ได้จากการทำงานของตัวอย่างโปรแกรมที่ ๑๕-๒ ถ้าป้อนข้อมูลเข้าแบบเดียวกับที่ป้อนให้โปรแกรมที่ ๑๕-๑ เพื่อให้ได้ผลลัพธ์ตามภาพผลลัพธ์ที่ ๑๕-๑

ตัวอย่างที่ ๑๕-๑ และ ๑๕-๒ ใช้เมธอด `read` ในการรับข้อมูลเข้า ซึ่งคืนค่าได้ทีละอักขระเท่านั้น ซึ่งไม่สะดวกนักในการใช้กับงานประยุกต์ทั่วไปซึ่งมักต้องการรับข้อมูลเข้าทีละชุดอักขระหรือเป็นสายอักขระมากกว่า เนื่องจากไม่มีเมธอดที่ทำงานลักษณะนี้ให้ใช้อย่างที่เราต้องการกับกระแสรับเข้ามาตรฐาน ดังนั้นเพื่อความสะดวกในการใช้งานในครั้งต่อๆ ไปเราจึงควรสร้างเมธอดสำหรับทำหน้าที่นี้ขึ้นไว้ใช้งาน

ตัวอย่างที่ ๑๕-๓ เป็นตัวอย่างแสดงเมธอดสำหรับอ่านข้อมูลจากกระแสรับเข้ามาตรฐานทีละบรรทัด บรรทัดที่ ๓ - ๕ เป็นเมธอดสำหรับแสดงข้อความทางกระแสส่งออกมาตรฐาน โดยรับพารามิเตอร์เข้ามาเป็นสายอักขระ แล้วนำสายอักขระนั้นส่งไปพิมพ์ด้วย `System.out.print` เราสร้างเมธอดนี้ขึ้นมาเพื่ออำนวยความสะดวกในการแสดงข้อความบอกผู้ใช้ บรรทัดที่ ๖ - ๒๑ เป็นเมธอด `getLine` ใช้สำหรับรับข้อมูลทางกระแสรับเข้ามาตรฐานทีละบรรทัดแล้วส่งค่าคืนออกมาเป็นสายอักขระ เมธอดนี้ดัดแปลงมาจากตัวอย่างที่ ๑๕-๒ โดยเขียนให้กระชับเข้าและแทนที่จะส่งข้อมูลออกทางกระแสส่งออกมาตรฐานเรานำข้อมูลไปเก็บในสายอักขระแทน เมื่อผู้ใช้เคาะ <Enter> เข้ามา เมธอดนี้จึงจะส่งค่ากลับคืนออกไปเป็นสายอักขระ บรรทัด ๒๕ เป็นการเรียกใช้เมธอด `prompt` ที่เราเขียนขึ้นใช้เอง บรรทัด๒๕ มีการเรียกใช้เมธอด `getLine` แล้วนำไปแสดงผลในบรรทัดที่ ๒๖

เมธอด `getLine` ใช้สำหรับรับข้อมูลที่เป็นตัวอักษร ตัวเลข และอักขระพิเศษต่างๆ แต่ในบางครั้งเราต้องการรับค่าเฉพาะตัวเลขเพื่อนำไปคำนวณหรือเปรียบเทียบกับข้อมูลตัวเลข เมธอด `getLine` คืนค่าเป็นสายอักขระซึ่งไม่สามารถนำไปคำนวณได้ในทันที ถึงแม้สายอักขระนั้นจะมีอักขระที่เป็นตัวเลขอยู่ภายในก็ตาม เนื่องจากตัวเลขนั้นเก็บเป็นรหัสของอักขระตามมาตรฐาน ASCII เราจำเป็นต้องแปลงอักขระให้เป็นข้อมูลประเภทตัวเลขเช่นประเภท `int` หรือ `double` เสียก่อน แล้วจึงจะนำไปคำนวณได้ เราสามารถแปลงตัวเลขที่อยู่ในสายอักขระให้เป็นข้อมูลแบบตัวเลขได้หลายวิธี ตัวอย่างที่ ๑๕-๓ นี้มีการแปลงข้อมูลจากประเภทสายอักขระให้เป็นประเภท `int` ในบรรทัดที่ ๒๔ มีการสร้างอ๊อบเจ็กต์ของคลาส `Integer` ให้ชื่อว่า `iw` โดยส่งพารามิเตอร์ไปให้ผู้สร้างเป็นสายอักขระ `line` ในบรรทัดที่ ๓๐ ทำการดึงเอาค่าที่เป็น `int` จากอ๊อบเจ็กต์ `iw` ออกมาใช้ด้วยเมธอด `intValue` แล้วจึงนำค่า `int` ที่ได้ไปทำการเปรียบเทียบข้อมูลในบรรทัด ๓๑

ตัวอย่างที่ ๑๕-๓ แสดงการสร้างเมธอดสำหรับอ่านข้อมูลจากกระแสรับเข้ามาตรฐานทีละบรรทัด

```
๑. import java.io.IOException;
๒. class Input3 {
๓.     public static void prompt( String msg ) {
๔.         System.out.print(msg);
๕.     }
๖.     public static String getLine() {
๗.         StringBuffer s = new StringBuffer(80);
๘.         try {
```

```
๙.              int c;
๑๐.             while ((c = System.in.read()) != -1) {
๑๑.                 if ( c == 13 ) // ignore CR
๑๒.                     continue;
๑๓.                 if ( c == 10 ) // is a Line Feed ?
๑๔.                     break;
๑๕.                 s.append((char)c);
๑๖.             }
๑๗.         } catch (IOException e) {
๑๘.             System.err.println( e );
๑๙.         }
๒๐.         return s.toString();
๒๑.     }
๒๒.     public static void main ( String args [] ) {
๒๓.         String line;
๒๔.         prompt("Please enter your name : ");
๒๕.         line = getLine();
๒๖.         System.out.println("Hi " + line);
๒๗.         prompt("How old are you? ");
๒๘.         line = getLine();
๒๙.         Integer iw = new Integer(line);
๓๐.         int age = iw.intValue();
๓๑.         if (age < 18)
๓๒.             System.out.println("You are so young");
๓๓.         else
๓๔.             if (age > 60)
๓๕.                 System.out.println("You are senior citizen");
๓๖.     }
๓๗. }
```

ภาพผลลัพธ์ที่ ๑๕-๓ เป็นตัวอย่างผลลัพธ์จากการทำงานของโปรแกรมตัวอย่างที่ ๑๕-๓ ถ้าป้อนข้อมูลเข้าเป็น Sombat

```
Please enter your name : Sombat
Hi Sombat
How old are you? 17
You are so young
```

ภาพผลลัพธ์ที่ ๑๕-๓ แสดงผลลัพธ์จากการทำงานของตัวอย่างที่ ๑๕-๓

ตัวอย่างที่ ๑๕-๔ เป็นตัวอย่างโปรแกรมสำหรับแปลงหน่วยความยาว โดยแปลงจากนิ้วไปเป็นเซ็นติเมตร โปรแกรมนี้รับข้อมูลจากแผงแป้นอักขระเข้าไปเป็นสายอักขระแล้วแปลงข้อมูลจากสายอักขระให้เป็นประเภท double ก่อนคำนวณแล้วแสดงผล โปรแกรมนี้มีเมธอด getLine ยกมาจากตัวอย่างที่แล้ว ส่วนการแปลง

ข้อมูลนั้นต่างจากตัวอย่างที่แล้วเนื่องจากคราวนี้ต้องการเปลี่ยนสายอักขระให้เป็นข้อมูลประเภท `double` จึงใช้คลาส `Double` ช่วยแปลงให้ วิธีการแปลงคล้ายกับตัวอย่างที่๑๕-๓ ต่างกันที่ใช้คลาสและประเภทของตัวแปรต่างกันเท่านั้น

ตัวอย่างที่ ๑๕-๔ โปรแกรมสำหรับแปลงขนาดความยาวหน่วยเป็นนิ้วให้เป็นเซ็นติเมตร

```
๑. import java.io.IOException;
๒. class Input4 {
๓.     public static String getLine() {
๔.         StringBuffer s = new StringBuffer(80);
๕.         try {
๖.             int c;
๗.             while ((c = System.in.read()) != -1) {
๘.                 if ( c == 13 ) // ignore CR
๙.                     continue;
๑๐.                if ( c == 10 ) // is a Line Feed ?
๑๑.                    break;
๑๒.                s.append((char)c);
๑๓.            }
๑๔.        } catch (IOException e) {
๑๕.            System.err.println( e );
๑๖.        }
๑๗.        return s.toString();
๑๘.    }
๑๙.    public static void main ( String args [] ) {
๒๐.        String line;
๒๑.        System.out.print("Please enter length in unit of inch : ");
๒๒.        line = getLine();
๒๓.        Double dw = new Double(line);
๒๔.        double length = dw.doubleValue();
๒๕.        System.out.println("Length in centimeter is " + length *
   2.54);
๒๖.    }
๒๗.}
```

ภาพผลลัพธ์ที่ ๑๕-๔ ตัวอย่างผลลัพธ์จากการทำงานของโปรแกรมตัวอย่างที่ ๑๕-๔

```
Please enter length in unit of inch : 34
Length in centimeter is 86.36
```

ตัวอย่างที่ ๑๕-๔ ถามข้อมูลความยาวเพียงครั้งเดียว เมื่อคำนวณและแสดงผลแล้วโปรแกรมจะจบการทำงานไปเลย ซึ่งอาจไม่สะดวกต่อการใช้งานของผู้ใช้นักถ้าผู้ใช้ต้องการแปลงค่าหลายตัว ทำให้เสียเวลาและยุ่งยากในการพิมพ์คำสั่งเพื่อเรียกโปรแกรมขึ้นมาทำงานหลายครั้ง ทางที่ดีโปรแกรมของเราควรที่จะสามารถคำนวณได้หลายเที่ยวจนกว่าผู้ใช้จะพอใจแล้วสั่งเลิกโปรแกรม หากต้องการให้โปรแกรมทำเช่นนี้ได้ควรให้

โปรแกรมทำงานเป็นวงวน สิ่งที่จะต้องพิจารณาต่อมาก็คือ ถ้าโปรแกรมทำงานเป็นวงวนแล้ว จะให้โปรแกรมจบได้อย่างไร เราสามารถให้โปรแกรมเลิกงานออกจากวงวนได้หลายวิธี เช่น

1. กำหนดจำนวนไว้ล่วงหน้าในโปรแกรม แล้วคอยนับจำนวนว่ารับข้อมูลเข้ามาครบหรือยัง
2. รับข้อมูลเข้าเป็นตัวบอกจำนวนแล้วจึงอ่านข้อมูลอื่นที่จะนำไปใช้จริง
3. ตรวจสอบการจบของข้อมูล ซึ่งอาจตรวจสอบได้จากค่าที่คืนกลับมาจากบางเมธอด หรือบางกรณีสามารถใช้วิธีดักจับสิ่งผิดปรกติที่บ่งบอกว่าจบข้อมูล
4. ใช้ค่าพิเศษของข้อมูลเป็นตัวบอก เช่นใช้ค่าศูนย์ หรือ 999 เป็นต้น แล้วคอยตรวจสอบว่าผู้ใช้ป้อนค่านี้เข้ามาหรือยัง

ตัวอย่างที่ ๑๕-๕ ดัดแปลงมาจากตัวอย่างที่ ๑๕-๔ ให้ทำงานเป็นวงวน โดยที่คอยตรวจสอบว่าความยาวที่ผู้ใช้ป้อนเข้ามาเป็นศูนย์หรือไม่ ถ้าเป็นศูนย์ถือว่าผู้ใช้ไม่ต้องการให้คำนวณอีกต่อไป จึงออกจากวงวนจบโปรแกรมไป

ตัวอย่างที่๑๕-๕ ปรับปรุงมาจากตัวอย่างที่ ๑๕-๔ ให้ทำงานเป็นวงวน

```
๑. import java.io.IOException;
๒. class Input5 {
๓.     public static String getLine() {
๔.         StringBuffer s = new StringBuffer(80);
๕.         try {
๖.             int c;
๗.             while ((c = System.in.read()) != -1) {
๘.                 if ( c == 13 ) // ignore CR
๙.                     continue;
๑๐.                if ( c == 10 ) // is a Line Feed ?
๑๑.                    break;
๑๒.                s.append((char)c);
๑๓.            }
๑๔.        } catch (IOException e) {
๑๕.            System.err.println( e );
๑๖.        }
๑๗.        return s.toString();
๑๘.    }
๑๙.    public static void main ( String args [] ) {
๒๐.        String line;
๒๑.        System.out.println("enter length of zero to terminate
   program");
๒๒.        while ( true ) {
๒๓.            System.out.print("Please enter length in unit of inch :
   ");
๒๔.            line = getLine();
```

```
๒๕.             Double dw = new Double(line);
๒๖.             double length = dw.doubleValue();
๒๗.             if (length == 0)
๒๘.                 break;
๒๙.             else
๓๐.                 System.out.println("Length in centimeter is " + length
    * 2.54);
๓๑.         }
๓๒.     }
๓๓. }
```

การส่งข้อมูลออกทางกระแสส่งออกมาตรฐาน

เราได้ส่งข้อมูลออกทางกระแสส่งออกมาตรฐานกันมาแล้วตั้งแต่ต้น วิธีการหนึ่งที่เราใช้ก็คือ ส่งออกด้วยข้อความสั่ง `System.out.print(…)` หรือ `System.out.println(…)` ในคลาส `System` มีตัวแปรสถิตอยู่หลายตัว out เป็นหนึ่งในตัวแปรสถิตเหล่านั้น มันเป็นตัวแปรอ้างอิงเก็บที่อยู่ของอ๊อบเจ็กต์ที่ทำหน้าที่ส่งข้อมูลออกทางกระแสส่งออกมาตรฐาน ตัวแปร `out` ประกาศไว้ในคลาส `System` ดังนี้

```
public static final PrintStream out
```

จะเห็นว่า `out` เป็นตัวแปรอ้างอิงไปยังอ๊อบเจ็กต์ของคลาส `PrintStream` และกำหนดให้เป็น `final` จึงจำเป็นต้องถูกกำหนดค่าเริ่มต้นให้ตั้งแต่ตอนสร้างตัวแปรนี้ ดังนั้นจึงต้องมีการสร้างอ๊อบเจ็กต์ของคลาส `PrintStream` มาก่อนจึงจะทราบที่อยู่ของมันได้ ซึ่งเท่ากับว่ามีการเตรียมอ๊อบเจ็กต์สำหรับทำหน้าที่ส่งข้อมูลออกไว้ก่อนแล้ว เราจึงสามารถเรียกใช้เมธอดต่างๆ ของคลาส `PrintStream` ผ่านตัวแปรอ้างอิง `out` ได้ทันที โดยที่เราไม่ต้องสร้างอ๊อบเจ็กต์เอง อ๊อบเจ็กต์ที่สร้างขึ้นนี้ได้เปิดทางไปสู่กระแสส่งออกมาตรฐานรอไว้แล้ว เราจึงส่งข้อมูลออกทางกระแสส่งออกมาตรฐานได้ง่ายไม่ต้องสร้างอ๊อบเจ็กต์และเปิดทางออกเองเหมือนอย่างที่จำเป็นต้องทำกับการส่งข้อมูลออกทางอื่นที่ไม่ใช่กระแสส่งออกมาตรฐาน

ข้อมูลที่ส่งออกด้วยคลาส `PrintStream` นั้นจะถูกเก็บค้างไว้ในที่พักก่อนจนกว่าจะมีการสั่งขึ้นบรรทัดใหม่หรือส่งตัวอักขระขึ้นบรรทัดใหม่ (`'\n'`) ไปให้ จึงจะทำการส่งออกไปยังอุปกรณ์ที่ทำหน้าที่ปลายทางในขณะนั้น คลาส `PrintStream` มีหลายเมธอดให้ใช้ ในที่นี้จะกล่าวถึงเฉพาะเมธอดที่มีโอกาสใช้บ่อย

เมธอดที่เราใช้กันมาแล้วคือเมธอด `print` และ `println` ซึ่งทำหน้าคล้ายกัน ต่างกันที่เมธอดหลังจะทำการพิมพ์และขึ้นบรรทัดใหม่ให้ทันที ในขณะที่เมธอดแรกไม่ขึ้นบรรทัดใหม่ให้เว้นเสียแต่ว่าจะมีการส่งตัวอักขระขึ้นบรรทัดใหม่ไปให้มันพิมพ์ เมธอดทั้งสองนี้มีการซ้ำชื่อเพื่อให้สามารถรับพารามิเตอร์ได้หลายชนิด เมธอดทั้งสองสามารถรับพารามิเตอร์ได้หนึ่งตัวจากประเภทใดประเภทหนึ่งต่อไปนี้คือ `boolean char char[] double float int long String` และ `Object` ยกเว้นเมธอด `println` จะไม่ส่งพารามิเตอร์ไปให้ก็ได้ซึ่งหมายถึงเราต้องการขึ้นบรรทัดใหม่เพียงอย่างเดียวโดยไม่ส่งข้อมูลอื่นใดไปพิมพ์ เมธอด `print` และ `println` จะแปลงข้อมูลประเภทต่างที่เราส่งไปให้ให้อยู่ในรูปของอักขระแล้วส่งเป็นไบต์ออกไปพิมพ์

ดังที่กล่าวมาแล้วว่าข้อมูลที่ส่งไปให้อ๊อบเจ็กทของคลาส `PrintStream` นั้น จะยังไม่ถูกส่งออกไปพิมพ์ในทันทีเสมอไป ข้อมูลจะถูกเก็บพักไว้ก่อนรอจนกว่าจะถูกสั่งให้ขึ้นบรรทัดใหม่ เราสามารถบังคับให้ส่งข้อมูลออกไปพิมพ์ในทันทีได้โดยใช้เมธอด `flush` เช่นถ้าต้องการบังคับให้พิมพ์ข้อมูลที่คั่งค้างอยู่ในที่พักของกระแสส่งออกมาตรฐานในทันทีเราสามารถสั่งได้ด้วยข้อความสั่ง `System.out.flush()`

ขณะที่ส่งข้อมูลออกอาจเกิดข้อผิดพลาดบกพร่องจนทำให้ไม่สามารถส่งข้อมูลออกไปก็เป็นได้ แต่คลาส `PrintStream` ไม่มีการโยนสิ่งผิดปรกติใดๆ ไปให้อ๊อบเจ็กต์ใดดำเนินการ ต่างจากกรณีของการรับเข้าด้วยคลาส `InputStream` ถ้ามีสิ่งผิดปรกติเกิดขึ้น จะมีการโยนสิ่งผิดปรกติ `IOException` ไปให้อ๊อบเจ็กต์จัดการหรือเราจะดักจับและจัดการเสียเองก็ได้ กรณีของคลาส `PrintStream` ถึงแม้จะไม่มีการโยนสิ่งผิดปรกติแต่ก็มีเมธอดให้ตรวจสอบว่าเกิดข้อผิดพลาดขึ้นหรือไม่ เมธอดนี้มีชื่อว่า `checkError` มีการประกาศไว้ดังนี้

```
public boolean checkError()
```

เมื่อเรียกเมธอดนี้มันจะทำการ `flush` ข้อมูลที่ค้างอยู่ในที่พักออกไป ถ้ามีการส่งข้อมูลออกไปไม่ได้เกิดขึ้นมันจะคืนค่าเป็น `true`

การเปลี่ยนทิศของกระแสรับเข้าและส่งออกมาตรฐาน

โดยปรกติแล้วข้อมูลที่ส่งออกทางกระแสส่งออกมาตรฐานจะแสดงผลทางจอภาพเป็นมาตรฐาน เราสามารถบังคับเปลี่ยนทิศทางของข้อมูลให้ไปลงแฟ้มหรือออกทางอุปกรณ์อื่นได้โดยไม่ต้องเปลี่ยนแปลงโปรแกรม ส่วนวิธีการว่าจะทำได้อย่างไรนั้นขึ้นอยู่กับแต่ละระบบปฏิบัติการ ในกรณีของ MS-DOS เราสามารถเปลี่ยนทิศทางได้โดยให้โปรแกรมอื่นควบคุม หรือเปลี่ยนที่บรรทัดคำสั่งของ MS-DOS ด้วยเครื่องหมายมากกว่า (>) ดังตัวอย่างเช่น

```
java Program1 > prn
```

ตัวอย่างนี้ส่งผลลัพธ์ที่ส่งออกทางกระแสส่งออกมาตรฐานของโปรแกรม `Program1` ไปที่เครื่องพิมพ์ คำ `prn` ในที่นี้เป็นชื่ออุปกรณ์ที่ทำหน้าที่เครื่องพิมพ์โดยปริยาย ถ้ามีการต่อพ่วงเครื่องพิมพ์หลายตัวเราอาจระบุชื่อทางออกไปยังเครื่องพิมพ์นั้นโดยตรง เช่นใช้คำ `lpt1` แทน `prn` ได้

ถ้าเราต้องการส่งข้อมูลจากกระแสส่งออกมาตรฐานลงแฟ้มก็สามารถทำได้เช่นเดียวกัน ตัวอย่างเช่น

```
java Program1 > outfile.txt
```

ตัวอย่างนี้ outfile.txt เป็นชื่อแฟ้มข้อมูลที่เราต้องการเก็บข้อมูลไว้ ถ้าชื่อนี้ไม่ไปตรงกับชื่ออุปกรณ์ใดๆ ที่กำหนดไว้ล่วงหน้าแล้วจะถือว่าเป็นชื่อแฟ้ม

การส่งข้อมูลออกทางกระแสข้อผิดพลาดมาตรฐาน

เราสามารถส่งข้อความออกทางกระแสข้อผิดพลาดมาตรฐานได้ทำนองเดียวกับการส่งออกทางกระแสส่งออกมาตรฐาน เนื่องจากใช้คลาส `PrintStream` เช่นเดียวกัน ต่างกันที่ออกกันคนละทางเท่านั้น สำหรับทางออกของกระแสข้อผิดพลาดมาตรฐานนั้นไม่สามารถบังคับเปลี่ยนทิศทางที่บรรทัดคำสั่งของ DOS ด้วยเครื่องหมายมากกว่าได้

เราสามารถส่งข้อความออกทางกระแสข้อผิดพลาดมาตรฐานโดยอาศัยตัวแปรอ้างอิง `err` ซึ่งกำหนดไว้ในคลาส `System` ทำนองเดียวกับตัวแปร `out` ของกระแสส่งออกมาตรฐาน ตัวแปร `err` มีการประกาศไว้ดังนี้

```
public static final PrintStream err
```

วิธีใช้เหมือนกับกระแสส่งออกมาตรฐานต่างกันแต่เพียงชื่อตัวแปรเปลี่ยนจาก `out` มาเป็น `err` เท่านั้น จึงไม่ขอกล่าวในรายละเอียดให้ซ้ำซ้อน