Chapter 2
Input / Output
Variables
Operators

Sample program Involving Input/Output

write a program to:
read a string from user
    a prompt should be used
echo it on screen.

Input a line of text:
my name is Joe Bloggs
Your input was: my name is Joe Bloggs
import java.io.*;
public class Echo {
    public static void main (String[] args)
        throws IOException {// must include this, more later
        // create a text input stream
        BufferedReader stdin = new BufferedReader
            (new InputStreamReader (System.in));
        String message;
        System.out.println("Input a line of text");
        message = stdin.readLine();
        System.out.println("Your input was: "+ message);
    }
}

3 Predefined Objects for Input/Output

System.out → PrintStream → Input a line
System.err → PrintStream → Error in Input
System.in → InputStream → String message = System.in.readLine();

? No Such method!
InputStream class provides methods for reading bytes. Java uses Unicode encoding (which allows most world languages to be handled) uses 2 bytes per character.

To read characters from System.in we need to chain a InputStreamReader object as in:

```java
InputStreamReader reader = new InputStreamReader(System.in);
char ch = reader.read();
```

However to read a line at a time we have to chain it to a BufferedReader object as in:

```java
BufferedReader console = new BufferedReader(new InputStreamReader(System.in));
String message = console.readLine();
```

**Chaining the objects**

```
console → BufferedReader → InputStreamReader → InputStream
```

- **InputStream**
  - byte readLine()
- **InputStreamReader**
  - char readLine()
- **BufferedReader**
  - String readLine()
Error Handling

Reading java docs for the BufferedReader method readLine() ...

```java
public String readLine() throws IOException
```

This method threatens to throw an exception if any errors encountered while reading.

When a method (within a class throws) an exception we can either

• Catch the exception and deal with it
• Pass it on by adding the phrase such as throws IOException as in

```java
class Echo {
    public static void main (String[] args) {
        throws IOException {
```

--

Write a program to add two numbers

where are the numbers coming from?
Are they whole numbers ? decimal part allowed ?
where should the output go to?

```java
Input the first integer number:
5
Input the second integer number:
7
The sum is: 12
```
import java.io.*;
public class AddingInts {
    public static void main(String[] args) throws IOException {
        BufferedReader stdin = new BufferedReader
            (new InputStreamReader(System.in));
        String string1, string2;
        int num1, num2, sum;
        System.out.println("Input an integer number");
        string1 = stdin.readLine();
        num1 = Integer.parseInt(string1);
        System.out.println("Input another integer number");
        string2 = stdin.readLine();
        num2 = Integer.parseInt(string2);
        sum = num1 + num2;
        System.out.print("The sum is: "+sum);
        System.out.println();
    }
}
How did we read an integer?

Create and chain the BufferedReader object to System.in

BufferedReader stdin = new BufferedReader(new InputStreamReader(System.in));

Read and store the user input in a String object

String string1;
string1 = stdin.readLine();

Be prepared to handle the exception thrown by readLine()

public static void main(String[] args) throws IOException {

To convert a String to integer use Integer.parseInt(.) as in

num1 = Integer.parseInt(string1);

Too tedious. Can we delegate this job to another class?

ConsoleReader class facilitates input

This user-created class is used in the book to facilitate input.

First create a ConsoleReader object associated System.in

ConsoleReader console =
    new ConsoleReader(System.in);

Subsequently to read an integer use:

int num1 = console.readInt();

Reading strings and doubles are just as easy!

String line = console.readLine();
double d = console.readDouble();
1st program rewritten using ConsoleReader

```java
import java.io.*;
public class Echo {
    public static void main (String[] args) {
        ConsoleReader console=new ConsoleReader(System.in);
        System.out.println("Input a line of text");
        String message = console.readLine();
        System.out.println("Your input was: " + message);
    }
}
```

The ConsoleReader class must be in the same directory

No need to chain
No need to handle exceptions
All these details are handled by class itself.

2nd program rewritten using ConsoleReader

```java
import java.io.*;
public class AddingInts {
    public static void main (String[] args) {
        ConsoleReader console=new ConsoleReader(System.in);
        int num1, num2, sum;
        System.out.println("Input an integer number");
        num1 = console.readInt();
        System.out.println("Input another integer number");
        num2 = console.readInt();
        sum = num1 + num2;
        System.out.println("The sum is: " + sum);
    }
}
```

No Chaining
No exception handling
No Strings required
No conversion to ints required
All details handled by class
import java.io.*;
public class ConsoleReader
{
    public ConsoleReader(InputStream inStream)
    {        reader = new BufferedReader(new InputStreamReader(inStream));
    }
    private BufferedReader reader;

    public String readLine()
    {        String inputLine = "";
        try {
            inputLine = reader.readLine();
        }
        catch(IOException e)
        {            System.out.println(e);
            System.exit(1);
        }
        return inputLine;
    }

    public int readInt()
    {        String inputString = readLine();
        int n = Integer.parseInt(inputString);
        return n;
    }

    public double readDouble()
    {        String inputString = readLine();
        double n = Double.parseDouble(inputString);
        return n;
    }
}
Coding Style

• Class Names start with a capital letter
• If there are several words start each word with uppercase letter
  
  *Echo, AddingInts, HelloWorld*
• For variables start with a lowercase letter
  
  *int anIntegerColor;*
  
  *String employeeName;*

Variables and Constants

How do we instruct the compiler we need a variable (declare) ?

```plaintext
int num1;    // reserves 4 bytes
double rate; // reserves 8 bytes
var_type varName; // general format
```

Can we initialize at declaration ?

```plaintext
int num1 = 100l;
double rate = 12.56 * 2;
// general format
var_type varName = expression;
```
Variables and Constants

What if I want the value to remain fixed?

```java
final int MAX_STUDENTS = 100;
final int TAX_BRACKET = 30000;
```

The keyword `final` has different meanings when applied to different things such as classes and methods.

Quiz: Which of the following are valid statements?

(a) `final double PI = 3.14;`  (b) `final PI;`  
   `PI = 3.14;`

(c) `final double PI = 3.1;`  
   `PI = 3.14;`

Use all uppercase letters for constants

Primitives and Objects

Primitives such as ints and doubles are created and manipulated directly using their names. (Though compiler uses their addresses it does not concern us)

```java
int x = 10;
x = x + 2;
```

Objects have no names and are manipulated indirectly using an object reference.

```java
Rectangle rect = new Rectangle(10,5,40,50);
rect.translate(20,40);
```
Primitive Data Types

Primitives in Java are byte, short, int, float, double, char and boolean. The first 4 are numeric types.

Some problems require us to handle whole numbers while others require floating point numbers (with decimal part).

What kind of variable would you use for:
- day, month, year?
- Temperature, weight(kg)?

Integers and Floating Point

- Integer numbers (using byte, short and int) are stored using exact representation
- Floating point numbers are stored separately using the mantissa and exponent. As the number of significant digits that can be stored is limited they cannot be represented exactly. Hence Floating point representations are only approximations.

\[ 27.56 = 2.756 \times 10^1 \quad \text{mantissa} = 2756 \quad \text{exponent} = 1 \]

- Most high-level languages don’t specify the sizes of their data types - impediment for portability.
- Java specifies the sizes precisely (independent of platform)
Java Numeric Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Size</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte</td>
<td>8 bits</td>
<td>-128</td>
<td>127</td>
</tr>
<tr>
<td>Short</td>
<td>16 bits</td>
<td>-32,768</td>
<td>32,767</td>
</tr>
<tr>
<td>Int</td>
<td>32 bits</td>
<td>-2,147,483,648</td>
<td>2,147,483,647</td>
</tr>
<tr>
<td>Float</td>
<td>32 bits</td>
<td>App. -3.4E+38</td>
<td>App. +3.4E+38</td>
</tr>
<tr>
<td>Double</td>
<td>64 bits</td>
<td>App. -1.7E+308</td>
<td>App. +1.7E+308</td>
</tr>
</tbody>
</table>

Typically use int for integers and double for floating point.

For large integers use long.

For small integers (day, month, year) you may use byte if memory is a constraint.

Data Types char and boolean

Most languages such as C, Pascal use 1 byte for char (8 bits) allowing only 256 distinct characters to be stored.

Java uses 2 bytes (Unicode) for characters allowing asian languages to be stored (up to 65536 values).

But most Operating Systems use 1 byte character - hence we have to convert them using classes such as InputStreamReader.

```java
char input = 'q';
char ch = '
'; // \n == new line \t == tab
```

boolean is used for storing true / false values

```java
boolean stop = false;
```
Strings

String name = new String(“Charles”);

is equivalent to: (strings are special)

String name = “Charles”;

What if I write the statement below next?

String name = “Charles Theva”;

A new String object will be created and it’s address will be assigned to name.
The old String object (at address 1000) will be marked for garbage collection.

Once a String object is created it cannot be changed (immutable object) in any way!

What will be the output?

import java.awt.Rectangle;
public class HelloRectangle { 
    public static void main(String[] args){
        Rectangle rect1 = new Rectangle(10,5,20,30);
        Rectangle rect2 = rect1;
        rect1.translate(20,30); // shifting
        rect2.translate(10,20);
        System.out.println(rect1);
    }
}

Output from the program

java.awt.Rectangle[x =__, y =__, width = 20, height = 30]
String Operations

String myName = "George";
the snapshot of memory is

0 1 2 3 4 5
G e O r g e
Note that the first index is 0, not 1.
Once a String object is created we can use its methods, such as

myName.charAt(2);    // returns 'o'
myName.indexOf('e');   // returns 1, it is the first 'e'

Comparing Strings

• the strings are compared character by character, until one character in s1 is less than its corresponding character in s2, then s1.compareTo(s2) < 0 is true

• if s1 is shorter than s2, and the characters in s1 are identical to the corresponding characters in s2, then s1.compareTo(s2) < 0 is true

Examples:

"George".compareTo("George") == 0 is true
"Georg".compareTo("George") < 0 is true
"George".compareTo("Georg") > 0 is true
"GEORGE".compareTo("George") < 0 is true
"   pam".compareTo("pam") < 0 is true
Operators

Arithmetic Operators + - * / %

What will be the output?

```java
int x = 10;
x = x + 5; // short form x += 5; (assignment operator)
x = x / 4; // short form x /= 4;
x = x % 2; // short form x %= 2;
System.out.println("Current value of x is " + x);
```

Casting

What will be the output?

```java
double ratio1, ratio2, ratio3;
int x = 10;
int y = 20;
ratio1 = x/y;
ratio2 = (double) (x / y);
ratio3 = (double) x / y;
System.out.println("ratio1 = " + ratio1);
System.out.println("ratio2 = " + ratio2);
System.out.println("ratio3 = " + ratio3);
```
Relational and Logical Operators

The relational operators <, <=, >, >= have the natural meaning yielding a boolean value (true/false).

To further manipulate these values Java provides the logical operators && (AND), || (OR), ^ (XOR), and ! (NOT). The values of these operations are given by the truth tables shown below.

| a | b | !a | a && b | a || b | a^b |
|---|---|----|--------|--------|-----|
| T | T | F  | T      | T      | F   |
| T | F | F  | F      | T      | T   |
| F | T | T  | F      | T      | T   |
| F | F | T  | F      | F      | F   |

What will be the output?

```java
class Ops {
    public static void main(String[] args) {
        int x = 10, y = 20, z = 15;
        boolean check1, check2, check3, check4;
        check1 = y > x;
        check2 = y > x && x > y;
        check3 = y > x || x > y;
        check4 = y > x && ! (x > y);

        System.out.println("check1 = " + check1);
        System.out.println("check2 = " + check2);
        System.out.println("check3 = " + check3);
        System.out.println("check4 = " + check4);
    }
}
```
Unary Operators (only one operand)

Unary minus -x

Operators ++ and -- (can be post or pre)

What will be the output?

```java
public class Ops {
    public static void main(String[] args) {
        int a = 10, p, q, r, s;
        p = a++;
        q = ++a;
        r = --a;
        s = a--;
        System.out.println("p = " + p);
        System.out.println("q = " + q);
        System.out.println("r = " + r);
        System.out.println("s = " + s);
        System.out.println("a = " + a);
    }
}
```

Operator precedence

<table>
<thead>
<tr>
<th>Category</th>
<th>Operators</th>
<th>Associativity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operations on references</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Unary</td>
<td>++, --, !, (type)</td>
<td>R to L</td>
</tr>
<tr>
<td>Multiplicative</td>
<td>*, /, %</td>
<td>L to R</td>
</tr>
<tr>
<td>Additive</td>
<td>+, -</td>
<td>L to R</td>
</tr>
<tr>
<td>Shift (bitwise)</td>
<td>&lt;&lt;, &gt;&gt;, &gt;&gt;&gt;</td>
<td>L to R</td>
</tr>
<tr>
<td>Relational</td>
<td>&lt;, &lt;=, &gt;, &gt;=, instanceof</td>
<td>L to R</td>
</tr>
<tr>
<td>Equality</td>
<td>==, !=</td>
<td>L to R</td>
</tr>
<tr>
<td>Bitwise AND</td>
<td>&amp;</td>
<td>L to R</td>
</tr>
<tr>
<td>Bitwise XOR</td>
<td>^</td>
<td>L to R</td>
</tr>
<tr>
<td>Bitwise OR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logical AND</td>
<td>&amp;&amp;</td>
<td>L to R</td>
</tr>
<tr>
<td>Logical OR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Conditional</td>
<td>? :</td>
<td>R to L</td>
</tr>
<tr>
<td>Assignment</td>
<td>=, +=, /=, %=, +=, -=</td>
<td>R to L</td>
</tr>
</tbody>
</table>
What’s wrong with this program?
Correct it!

```java
public class Ops {
    public static void main(String[] args) {
        double a = 10.0;
        double b = 20.0;
        double aver = a + b / 2;
        System.out.println("average of a and b = " + aver);
    }
}
```

Errors in Chapter 2

Page 41: program (lower half)
```java
message = ConsoleReader.readline(); message = console.readLine();
```

Page 42: program readDouble() method
```java
int n = Integer.parseInt(inputString);
double n = Double.parseDouble(inputString);
```
(The throws clause in main() can be removed for both programs)