Contents

1. Introduction
2. Types and Variables
3. Statements and Control Flow
4. Reading Input
5. Classes and Objects
6. Arrays
7. Methods
8. Scope and Lifetime
9. Utility classes
10. Introduction to Object-Oriented Analysis and Design
Chapter 2: Types and Variables

After this chapter you will be able to:

- Declare variables using types
- Describe the syntax and layout of Java
- Write a simple Java program
Key to Symbols

Crystal ball - indicates a topic that is covered in more detail later in the course

Elephant - indicates an important point not to forget

Bug - indicates an erroneous program

Check - indicates a correct program
# Integer types

<table>
<thead>
<tr>
<th>Type</th>
<th>Storage</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>4 bytes</td>
<td>-2,147,483,648 to 2,147,483,647</td>
</tr>
<tr>
<td>short</td>
<td>2 bytes</td>
<td>-32,768 to 32,767</td>
</tr>
<tr>
<td>long</td>
<td>8 bytes</td>
<td>-9,223,372,036,854,775,808L to 9,223,372,036,854,775,807L</td>
</tr>
<tr>
<td>byte</td>
<td>1 byte</td>
<td>-128 to 127</td>
</tr>
<tr>
<td>Type</td>
<td>Storage</td>
<td>Range</td>
</tr>
<tr>
<td>----------</td>
<td>---------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>float</td>
<td>4 bytes</td>
<td>+/- 3.40282347E+38F</td>
</tr>
<tr>
<td>double</td>
<td>8 bytes</td>
<td>+/- 1.79769313486231570E+308</td>
</tr>
</tbody>
</table>
## Character Type

<table>
<thead>
<tr>
<th>Type</th>
<th>Storage</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>char</td>
<td>2 bytes</td>
<td>\u0000 to \uFFFF</td>
</tr>
</tbody>
</table>

- The 2-byte Unicode allows 65,536 characters unlike ASCII, which is a 1-byte code allowing only 255 characters.
- Special characters:
  - \b backspace
  - \t tab
  - \n linefeed
  - \r carriage return
  - \" double quote
  - \' single quote
  - \\ backslash
## Boolean Type

<table>
<thead>
<tr>
<th>Type</th>
<th>Storage</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>1 bit</td>
<td>true or false</td>
</tr>
</tbody>
</table>
Questions

• What is the meaning of a type?
• If $x$ is an integer variable then what can we say about $x$?
Answer

• A variable's type implies **both**:  
  – the legal operations that may be applied to the variable  
  – the range of values that may be stored in the variable

• If $x$ is an integer then:  
  – we can apply arithmetic operations to $x$  
  – we can store integer values in $x$

Types have similar implications for object variables.
Variable Declarations

• Syntax: \textit{type \ var [= value] [, \ var [= value]]*;}

• Examples

  int \ x;
  char \ c = 'h';
  boolean \ b = true;
  short \ y = 3, \ z = 10;
String Literals

• Double quotes are used to create strings

"Hello world"
"This is a very long string"

Variables can also be declared using a String type
Floating Point Literals

• By default floating point literals are assigned type **double**

```java
double d = 3.14;
```

Literal has type **double**
Incompatible Types

• **Question:** What is wrong?

```c
float f = 3.14;
```

• **Answer:** Both sides of `=` must have the same type (compile-time error)

• In this case `f` has type `double` but `3.14` has type `double`
Compatible Types

- Adding `f` to the end of `3.14` indicates a `float` literal

```c
float f = 3.14f;
```

Literal has type `float`
First Java Program

```java
public class Hello {
    public static void main (String [] args) {
        System.out.println("Hello world");
    }
}
```

Output

```
Hello world
```

- Every program has a routine named `main`
- `main` is always placed inside a class that has an arbitrary name
Meaning of Braces

```java
public class Hello {
    public static void main (String [] args) {
        System.out.println("Hello world");
    }
}
```
Whitespace Not Significant

```java
public class Hello { public static void main (String [] args) {System.out.println("Hello world");}}
```

Output  Hello world
Output

- `System.out.println` prints its argument on the screen.
- Examples:
  ```java
  System.out.println("Hello");
  System.out.println(33);
  System.out.println('a');
  ```

- `System.out.print` does the same without outputting a newline:
  ```java
  System.out.print("Hello");
  ```
Variables in Programs

```java
public class Printx {
    public static void main (String [] args) {
        int x = 100;
        System.out.print ("x's value = ");
        System.out.println(x);
    }
}
```

Output  x's value = 100

• A variable may be declared anywhere inside main
Assignment

```java
public class Printx {
    public static void main (String [] args) {
        int x = 100;
        System.out.print ("x's value = ");
        System.out.println(x);
        x = 200;
        System.out.print ("x's value = ");
        System.out.println(x);
    }
}
```

- Program output: ?
Arithmetic Operators

• +,-,*,/

• % (integer division remainder)

• Example:

\[ x = 2 * 10 / 5 \% 3; \]

• What value is in \( x \)?
Boolean Operators

• == (equality test)
• != (not equals)
• >, <, <=, >=

• Examples:
  m == n
  a != c
  x >= y
More Boolean Operators

- `&&` and
- `||` or
- `!` negation

**Examples**

```
((20 / 4) == 5) || (7 >= 10)
```

```
((10 * 3) == 30) && ((2 + 3) <= 6)
```

**Are these true or false?**
Comments

public class Printx {
    // this comments rest of line
    public static void main (String [] args) {
        /* this is a
           multi-line comment */
        int x = 100;
        System.out.print ("x's value = ");
        System.out.println(x);
    }
}

• Java has two styles of comments
Java File Naming

public class Hello {
    public static void main (String [] args) {
        System.out.println("Hello world");
    }
}

- All Java programs must be saved in a file that has the class name as a prefix, and .java as the suffix.

Java is case sensitive (i.e., Hello is not hello)
It's Exercise Time