Lecture 2 — Variables, Ifs, and Loops

CITS2005 Object Oriented Programming

Department of Computer Science and Software Engineering
University of Western Australia

Contents

- See Chapter 1 of the textbook (and a bit of Chapter 3)
- Variables, types, and operators (arithmetic + relational)
- `if` statements, code blocks
- `while`, `for`, and `do-while` loops

MyNumber

```
public class MyNumber {
    public static void main(String[] args) {
        // Declares a variable
        int myNumber;
        myNumber = 5*2 + 1; // Assigns the variable a value
        System.out.println("myNumber is: " + myNumber);
    }
}
```

- Let's run the code
- Prints "`myNumber is:  11`"

Variable Declaration

```
public class MyNumber {
    public static void main(String[] args) {
        // Declares a variable
        int myNumber;
        myNumber = 5*2 + 1; // Assigns the variable a value
        System.out.println("myNumber is: " + myNumber);
    }
}
```

- A variable (`myNumber`) is declared
- Variables in Java have types and are declared as follows: `type name;`
- The type `int` means it holds *integer* values
- (e.g., 1, -5, 100, -5134, 15)

Variable Assignment

```java
public class MyNumber {
    public static void main(String[] args) {
        // Declares a variable
        int myNumber;
        myNumber = 5*2 + 1; // Assigns the variable a value
        System.out.println("myNumber is: " + myNumber);
    }
}
```

- We *assign* `myNumber` a value
- This is done using the = operator
- Because `myNumber` is an `int`, Java needs the right-hand side to be an `int`
- The `//` parts are *comments*

Variable Assignment and Type Errors

```
public class MyNumber2 {
    public static void main(String[] args) {
        String myString = "hi";
        int myNumber;
        myNumber = 5*2 + 1;
        myNumber = myString; // causes a type error
        System.out.println("myNumber is: " + myNumber);
    }
}
```

- Java lets you declare and assign a variable in a single statement: `String myString = "hi";`
- `MyNumber2.java:6: error: incompatible types: String cannot be converted to int`

More Types

```
public class MyNumber2 {
    public static void main(String[] args) {
        String myString = "hi";
        int myNumber;
        myNumber = 5*2 + 1;
        myNumber = myString; // causes a type error
        System.out.println("myNumber is: " + myNumber);
    }
}
```

- Recall our method declaration: `public static void main(String[] args)`
- `void` is the *return value*
- `void` is a special type that means "there is no return value"
- `String[] args` is a special kind of variable called a *parameter*

Arithermic Operators

- Java supports many integer arithmetic operators other than addition

`+`	Addition	`5+3 (=8)`
`-`	Subtraction	`5-3 (=2)`
`*`	Multiplication	`5*3 (=15)`
`/`	Division	`5/3 (=1), 6/3 (=2)`
`%`	Modulus	`5%3 (=2), 6%3 (=0)`

Floating Point Numbers

- Non-integer numbers are represented by the type `double` (or sometimes `float`)
- These are numbers with a decimal place: 0.02342, 1.333333
- They support similar operators (`+`, `-`, `*`, `/` `%`), but they work differently

Floating Point Numbers

```
public class DoubleExample {
    public static void main(String[] args) {
        int x = 5;
        double y = 5.0;
        x = x / 2;
        y = y / 2;
        System.out.println("x = " + x);
        System.out.println("y = " + y);
    }
}
```

```
x = 2
y = 2.5
```

If Statement

```
public class IfExample {
    public static void main(String[] args) {
        if (2 < 4)
            System.out.println("2 is less than 4");
        if (2 > 4)
            System.out.println("Impossible!");
    }
}
```

- `if` statements are conditional statements that control the flow of a program
- `if(`*`condition`*`)` *`statement;`*
- The condition must be a *boolean expression*

Boolean Operators

- Conditions are usually constructed using relational operators

`==`	Equal to	`2==3 (false)`
`!=`	Not equal	`2!=3 (true)`
`<`	Less than	`2<3 (true)`
`>`	Greater than	`2>3 (false)`
`<=`	Less than or equal	`2<=3 (true)`
`>=`	Greater than or equal	`2>=3 (false)`

If Block

```
public class IfBlockExample {
    public static void main(String[] args) {
        int a = 2, b = 3, c;
        c = 4;
        if (a+b > c) {
            c = a+b;
            a = b;
        }
        System.out.println("a = " + a);
        System.out.println("b = " + b);
        System.out.println("c = " + c);
    }
}
```

If Block

```
int a = 2, b = 3, c;
```

- Many declarations can be made at once using the comma

```
if (a+b > c) {
    c = a+b;
    a = b;
}
```

- $\{\ldots\}$ can group multiple statements into a *block*

Statements end with ;

```
if (a+b > c) {
    c = a+b; a = b;
}
```

- This code is the same
- White space in Java is not important
- This is why statements always end with a ;

Formatting of statements

```
// Same line
if (a+b > c) c = a+b;
// Different lines
if (a+b > c)
    c = a+b;
```

- Both are the same
- White space in Java is not important

if ... else if ... else

```
public class IfElseIfElse {
    public static void main(String[] args) {
        int a = 3;
        if (a == 1) {
            System.out.println("a is 1");
        } else if (a == 2) {
            System.out.println("a is 2");
        } else if (a == 3) {
            System.out.println("a is 3");
        } else {
            System.out.println("a is not 1, 2, or 3");
        }
    }
}
```

- An `if` can be followed by 0 or more `else if` statements, then 0 or 1 `else` statements

if ... else if ... else ordering

```
public class IfElseIfElse2 {
    public static void main(String[] args) {
        int a = 3;
        if (a > 2) {
            System.out.println("a is greater than 2");
        } else if (a > 1) {
            System.out.println("a is greater than 1");
        } else {
            System.out.println("a is not greater than 1");
        }
    }
}
```

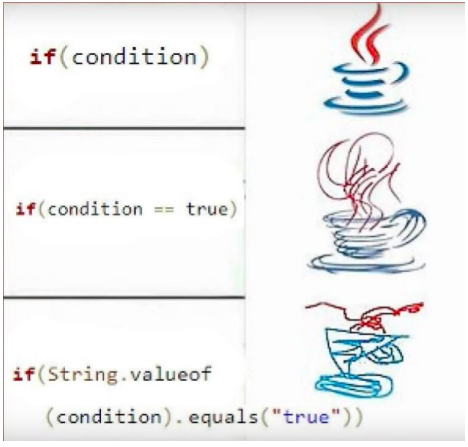
- Only prints `a is greater than 2`
- Be careful, these are checked in order, then all the rest get skipped!

if ... else if ... else ordering

```
public class IfElseIfElse3 {
    public static void main(String[] args) {
        int a = 3;
        if (a > 2) {
            System.out.println("a is greater than 2");
        }
        if (a > 1) {
            System.out.println("a is greater than 1");
        }
    }
}
```

- This prints both

Mid-lecture break



while loop

```
public class WhileLoop {
    public static void main(String[] args) {
        int iteration = 1;
        while (iteration <= 5) {
            System.out.println("iteration #" + iteration);
            iteration = iteration + 1;
        }
    }
}
```

- The `while` loop controls the flow a program
- It works like an `if` statement that loops

for loop

```
public class ForLoop {
    public static void main(String[] args) {
        for (int i = 1; i <= 5; i += 1)
            System.out.println("iteration #" + i);
    }
}
```

- `i += 1` is a special operator that means `i = i+1`
- The `for` loop makes loops that iterate over ranges easier
- Designed as a shorthand for replacing a `while` loop

i++

```
public class ForLoop2 {
    public static void main(String[] args) {
        for (int i = 1; i <= 5; i++)
            System.out.println("iteration #" + i);
    }
}
```

- `i++` is more idiomatic than `i += 1`
- It *almost* means the same thing
- They are the same unless you write code like `x = y+=1` (yes this is valid Java)

i++

```
public class IPlusPlus {
    public static void main(String[] args) {
        int i = 1;
        int x = i++;
        System.out.println("i = " + i + ", x = " + x);
        int y = i+=1;
        System.out.println("i = " + i + ", y = " + y);
    }
}
```

- What will happen if we run this code?

for-each loop

```
public class ForEachLoop {
    public static void main(String[] args) {
        for (String arg : args)
            System.out.println(arg);
    }
}
```

- Iterates though the contents of args, which is an array
- *syntax-sugar* for loop over a collection
- `for (Type element : collection) statement;`

do-while loop

```
public class DoWhileLoop {
    public static void main(String[] args) {
        int iteration = 1;
        do {
            System.out.println("iteration #" + iteration);
            iteration++;
        } while (iteration <= 5);
    }
}
```

- `do {...} while(...);` is the syntax for a `do-while` loop
- Checks to condition after an iteration
- Means at least 1 iteration is always executed
- Why would you want this?

Guess the Number

```
import java.util.Scanner;

public class GuessTheNumber {
    public static void main(String[] args) {
        int secretNumber = 9;
        System.out.println("I am thinking of a number between 1 and 10. Guess what it is.");
        Scanner scanner = new Scanner(System.in);
        int guess;
        do {
            guess = scanner.nextInt();
            if (guess != secretNumber)
                System.out.println("Nope, try again.");
        } while (guess != secretNumber);
        System.out.println("You got it!");
    }
}
```

- Lots to understand here! First, let's demo the code.

Guess the Number

```
import java.util.Scanner;

public class GuessTheNumber {
    public static void main(String[] args) {
        int secretNumber = 9;
        System.out.println("I am thinking of a number between 1 and 10. Guess what it is.");
        Scanner scanner = new Scanner(System.in);
        int guess;
        do {
            guess = scanner.nextInt();
            if (guess != secretNumber)
                System.out.println("Nope, try again.");
        } while (guess != secretNumber);
        System.out.println("You got it!");
    }
}
```

- `Scanner` is part of the *Java Class Library* and must be `imported` (more of these later)

Guess the Number

```
import java.util.Scanner;

public class GuessTheNumber {
    public static void main(String[] args) {
        int secretNumber = 9;
        System.out.println("I am thinking of a number between 1 and 10. Guess what it is.");
        Scanner scanner = new Scanner(System.in);
        int guess;
        do {
            guess = scanner.nextInt();
            if (guess != secretNumber)
                System.out.println("Nope, try again.");
        } while (guess != secretNumber);
        System.out.println("You got it!");
    }
}
```

- `new` is used to create an object (more on this later)
- `.nextInt()` *calls* a *method* of `scanner`

Guess the Number

```
import java.util.Scanner;

public class GuessTheNumber {
    public static void main(String[] args) {
        int secretNumber = 9;
        System.out.println("I am thinking of a number between 1 and 10. Guess what it is.");
        Scanner scanner = new Scanner(System.in);
        int guess;
        do {
            guess = scanner.nextInt();
            if (guess != secretNumber)
                System.out.println("Nope, try again.");
        } while (guess != secretNumber);
        System.out.println("You got it!");
    }
}
```

- `do-while` is useful here since the user needs to make a first guess