COMP1409: Introduction to Software Development I

Week 6

Mike Mulder

Agenda

- Quiz
  - Quiz 5
  - Review Answers
- Review
  - Java Classes and Syntax
  - Lab 5 Solution Review
- Lesson 6
  - Object Interaction
  - Abstraction/Modularization
  - Composition
  - External Method Calls
- Lab 6

Quiz 4

Closed book, laptop, phone, etc.

You have a maximum of 20 minutes to complete

Raise your hand when you are done, and I will retrieve your paper

We will review the answers afterwards

```
public class Point {
    private static final int MIN_X = 0;
    private static final int MAX_X = 100;
    private int x = 0;
    private int y = 0;

    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public int getX() {
        return this.x;
    }

    public void setX(int x) {
        this.x = x;
    }

    private isXValid(int x) {
        boolean isValid = false;
        if (x >= MIN_X && x <= MAX_Y) {
            isValid = true;
        }
        return isValid;
}
```

A – Class Name (and Visibility)

B - Constants

C – Instance Variables (including visibility and type)

D – Constructor (including visibility and parameters)

E – Methods (Today's Class)

F – Accessor/Getter/Get Method

G – Mutator/Setter/Set Method

H – Private Method

I – Local Variable

```
...
public void printDetails() {
    System.out.println("Details");
}

public void printDetails(String name) {
    System.out.println("Details: " + name);
}

private boolean isValidOperation(String operation) {
    boolean isValid = false;
    switch (operation) {
        case "*":
        case "%":
        case "+":
        case "-":
            isValid = true;
            break;
         default:
             isValid = false;
             break;
    }
    return isValid;
}
```

A

B

A – Overloaded Methods

B – Switch

Equivalent If Statement:

```
if (operation.equals("*") ||
    operation.equals("%") ||
    operation.equals("+") ||
    operation.equals("-")) {
    isValid = true;
} else {
    isValid = false;
}
```

```
...

private void printAge() {
    int currAge = 0;
    int currYear = this.birthYear;

    while (currAge < this.age) {
        System.out.print("Person in " + currYear +
             " was " + currAge + " year(s) old.");
        currAge++;
        currYear++;
    }
}

...
```

A

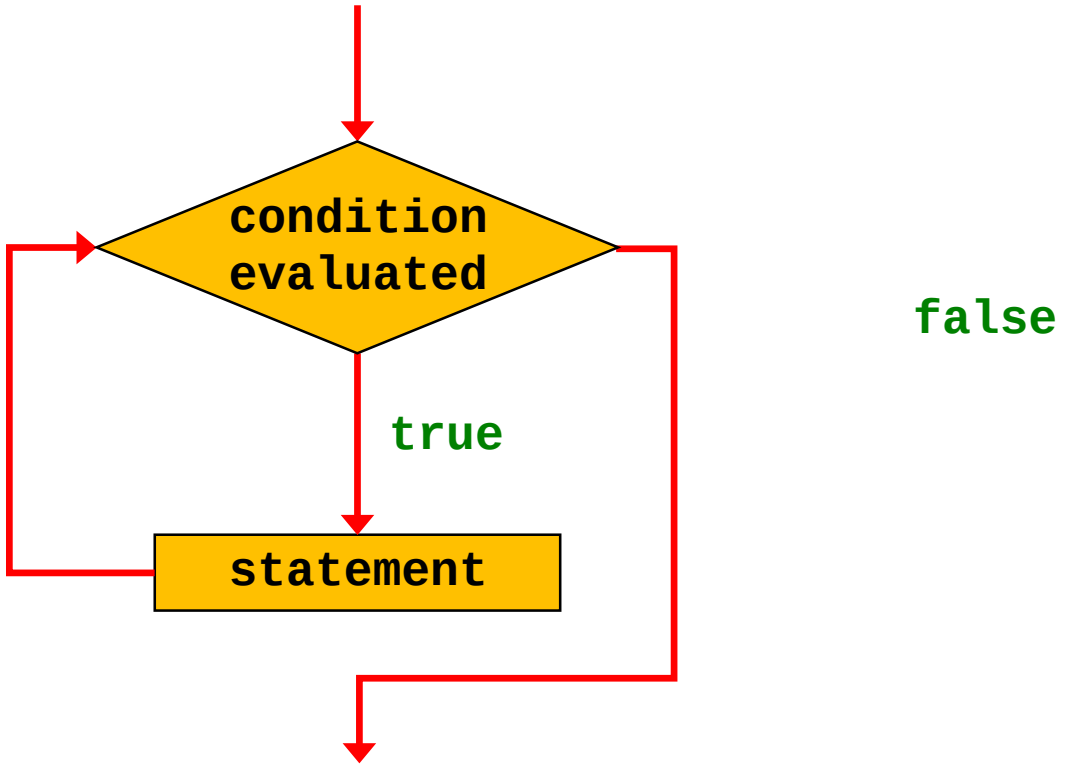
B

C

A – Condition – Logical and Relational Operations

B – Statements

C – Increment Index Variables (Prevent "Infinite Loops")

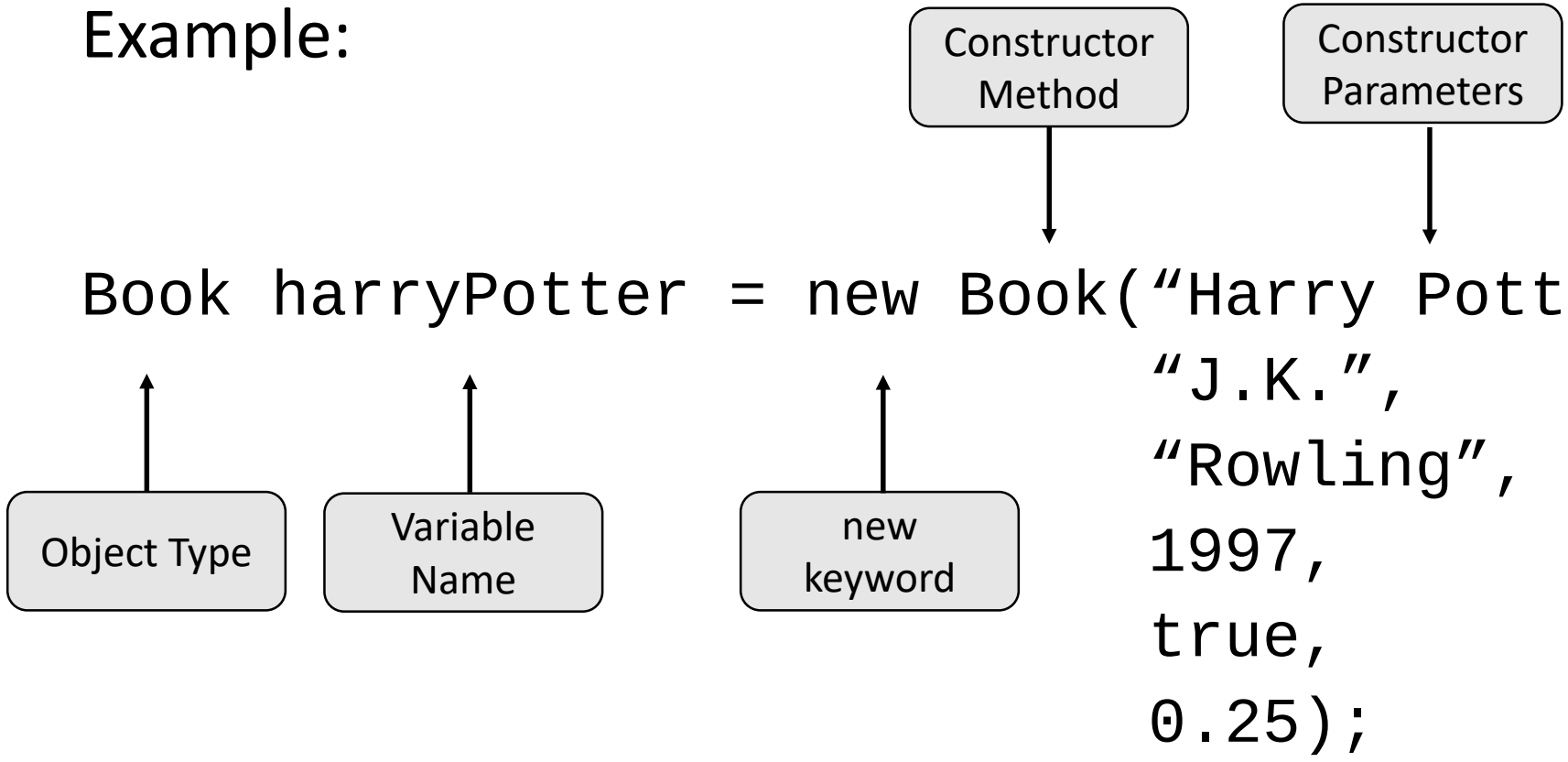


Arithmetic operators

+	addition
-	subtraction
*	multiplication
/	division
%	modulus (remainder after division)
++	increment
--	decrement
+=	add then assign
-=	subtract then assign

Creating an Instance of a Class (Review)

Example:




```
Book harryPotter = new Book(“Harry Potter”,
                            “J.K.”,
                            “Rowling”,
                            1997,
                            true,
                            0.25);
```

Lab 5

Review Sample Solutions

- Person.java
- Math.java

5B - Review Alternate Solution – Static Methods

Assignments

Assignment 1 will be marked early next week.

Assignment 2 will be posted to D2L before next week's class.
Due before the Week 9 class.

Learning Outcomes: Lesson 6

- Object interaction
- Abstraction
- Modularization
- Composition: classes as data types
- External method calls

Abstraction and modularization

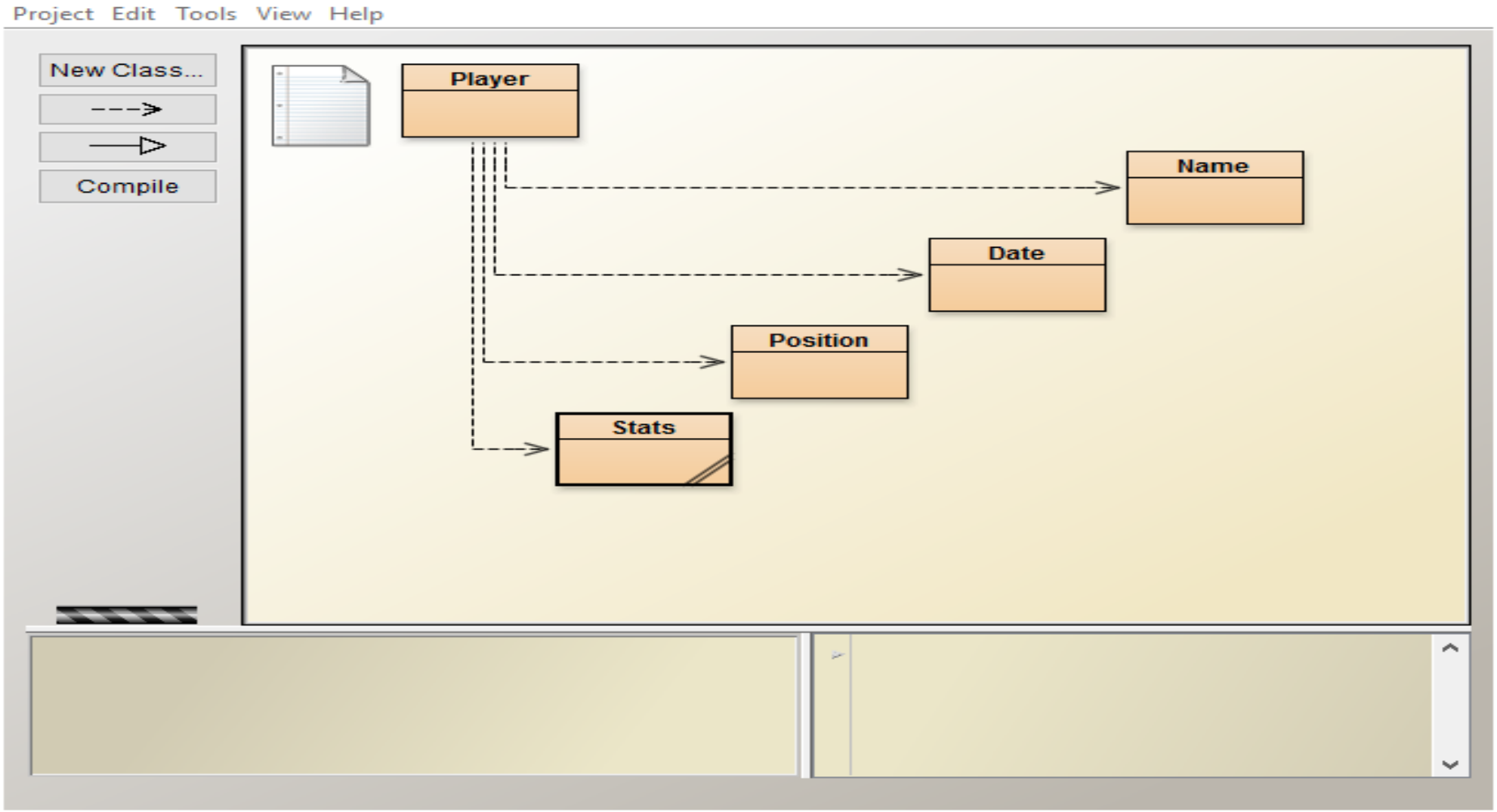
- Abstraction is the ability to ignore details of parts to focus attention on a higher level of a problem

- Modularization is the process of dividing a whole into well-defined parts, which can be built and examined separately, and which interact in well-defined ways.

Object interaction

- A java program creates many objects
- They interact with each other by invoking each other's methods
- Objects also contain other objects.

An example: Player



Abstraction and modularization

- Player is modular – composed of name, position, stats, and year-drafted classes

```
public class Player
{
  private Name        name;
  private Position    position;
  private Stats       stats;
  private Date        dateDrafted
```

Abstraction and modularization

- Player uses abstraction – we work with the components without worrying about their details
- The Player class (below) is using a Name method on the (private) name object:

```
public String getFullName()
{
    return name.getFullName();
}
```

Classes as data types

- Anything that is not a primitive data type in Java, is an object type
- We have worked with String objects
- Now we will work with other objects

```
private Name        name;
private Position    position;
private Stats       stats;
private Date        yearDrafted
```

Creating objects

- An object is created by using the Java key word ‘new’ and calling the object’s constructor
- `new Player()` creates an object from the Player class
- When this object is on the object bench in BlueJ, it has a name, e.g. player1

Creating objects

- When creating an object in a class, we first declare a variable and assign the new object to that variable

```
Player player1 = new Player();
```

- The data type is `Player`
- The variable name is `player1`

Calling methods

- We have created an object:

```
Player player1 = new Player();
```

- Now we can call **public** methods of the Player object, e.g.

```
player1.getName()
player1.printDetails()
```

Calling methods

- We have created an object:

```
Player player1 = new Player(“tiger”, “woods”);
```

- Now we can call methods of the Player object, e.g.

```
player1.setNumberOfWins(79)
player1.getName()
player1.printDetails()
```

Calling methods

Return values can be assigned to instance or local variables, e.g.

```
String name = player1.getName()
this.fullName = player1.getName()
```

Return values can be returned from your methods, e.g.

```
public String getPlayerName() {
    return this.player1.getName();
}
```

Composition

Classes that contain (or are composed of) instances of other classes.

This is one of the fundamental principles of Object Oriented Programming. It enables re-use and test-ablility.

Most of the software you build in real-life is composed of other classes built by your team or 3rd parties (i.e., open source)

Lab 6

Lab 6A

- You will need to load two classes into BlueJ (download from D2L: Week 6 -> NumberDisplay and ClockDisplay. This is also a DayDisplay class with the constants already defined that you can load into BlueJ.

- You will use these existing classes in the DayDisplay class you will create for the lab.

Lab 6B

- Due Thursday at Midnight.