Some miscellaneous concepts

Static, Javadoc and Calculated Data

Produced by: Dr. Siobhán Drohan
Mairead Meagher

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

Department of Computing and Mathematics
http://www.wit.ie/

Topic List

- Static Variables
- Static Methods
- Javadoc
- Storing calculated data

Instance vs Static (Class) Variables

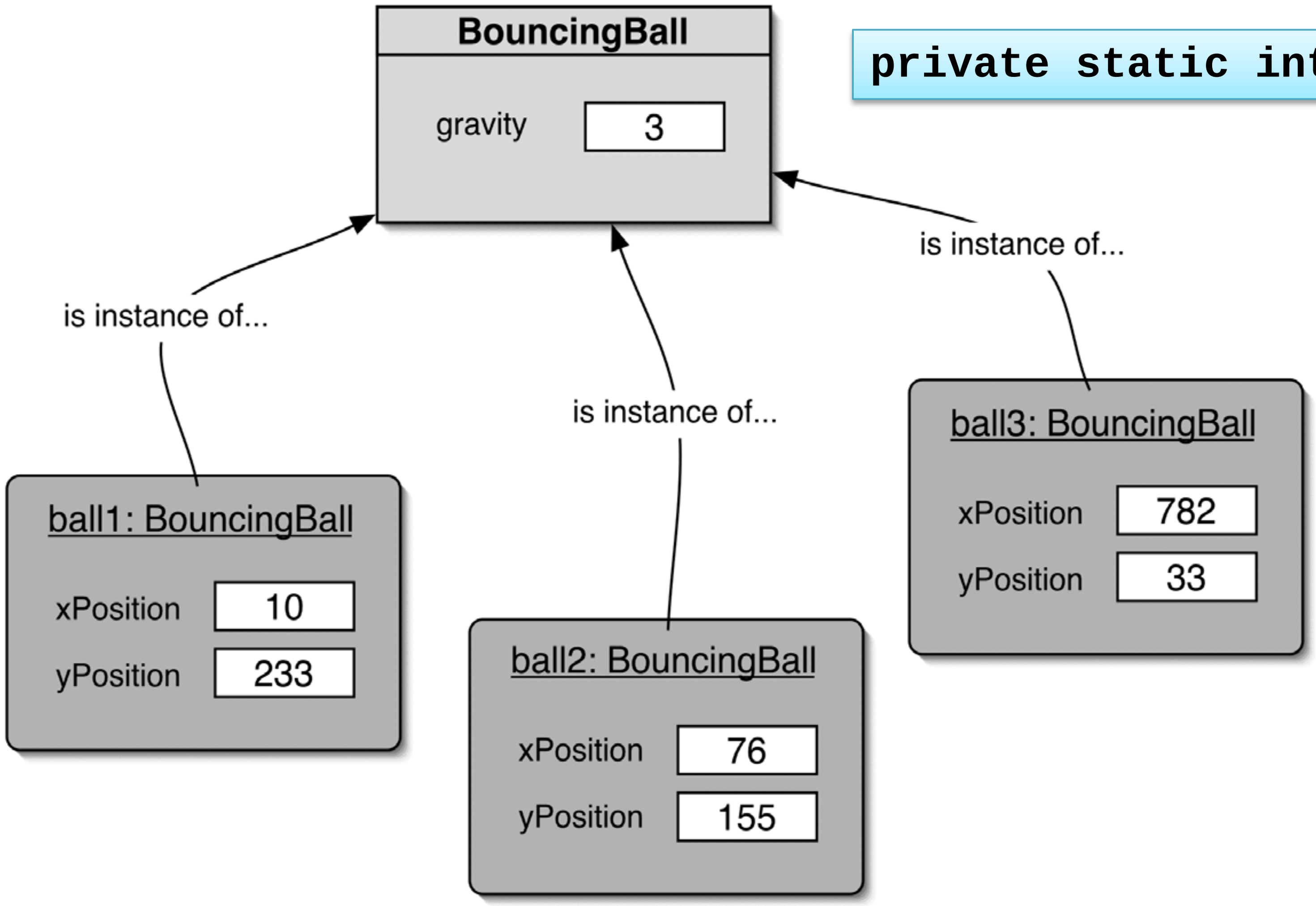
- When a number of objects are created from the same class blueprint, they each have their own distinct copies of *instance variables*.

- Sometimes, you want to have variables that are common to all objects. This is accomplished with the static modifier.

- Fields that have the static modifier in their declaration are called *static fields* or *class variables*.

https://docs.oracle.com/javase/tutorial/java/javaOO/classvars.html

Instance vs Static (Class) Variables



```
private static int gravity;
```

Constants

```
private static final int GRAVITY = 3;
```

- **`private`**: access modifier, as usual
- **`static`**: class variable
- **`final`**: constant (cannot change the value). Naming standards for final fields is all capitals.

Topic List

- Static Variables
- Static Methods
- Javadoc
- Storing calculated data

Static Methods

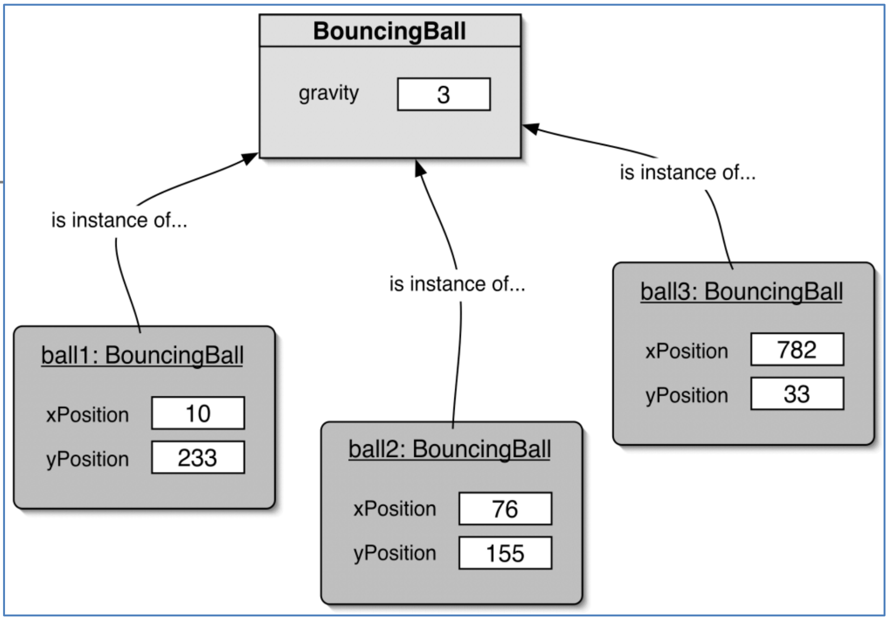
- Java supports static methods as well as static variables.

- Static methods, which have the static modifier in their declarations, should be invoked with the class name, without the need for creating an instance of the class, as in

```
ClassName.methodName(args)
```

Static Methods

A common use for static methods is to access static fields.



For example, we could add a static method to the BouncingBall class to access the gravity static field:

```
public static int getGravity()
{
    return gravity;
}
```

Topic List

- Static Variables
- Static Methods
- Javadoc
- Storing calculated data

Writing class documentation

- Your own classes should be documented the same way library classes are.
- Other people should be able to use your class without reading the implementation.
- Make your class a 'library class'!

Elements of documentation

Documentation for a class should include:

- the class name
- a comment describing the overall purpose and characteristics of the class
- a version number
- the authors' names
- documentation for each constructor and each method

Elements of documentation

The documentation for each constructor and method should include:

- the name of the method
- the return type
- the parameter names and types
- a description of the purpose and function of the method
- a description of each parameter
- a description of the value returned

javadoc

- The comment start symbol must be of this format in order to be recognised as a javadoc comment:

  /**

- Such a comment immediately preceding the:
  - class declaration is read as a class comment.
  - method signature is read as a method comment.

- Other special key symbols for formatting documentation include:

  @version
  @author
  @param
  @return

javadoc

Class comment:

```
/**
 * The Responder class represents a response
 * generator object. It is used to generate an
 * automatic response.
 *
 * @author     Michael Kölling and David J. Barnes
 * @version    1.0  (30.Mar.2006)
 */
```

javadoc

Method comment:

```
/**
 * Read a line of text from standard input (the text
 * terminal), and return it as a set of words.
 *
 * @param  prompt  A prompt to print to screen.
 * @return A set of Strings, where each String is
 *         one of the words typed by the user
 */
public HashSet<String> getInput(String prompt)
{
   ...
}
```

Topic List

- Static Variables
- Static Methods
- Javadoc
- Storing calculated data

The danger lurking within!

Calculated data

```
public class Employee
{
    private double salary;
    private double deductions;
    private double netSalary;
    :
    :
    public void calculateNetSalary()
    {
        netSalary = salary – deductions;
    }
    public void setSalary(double salary)
    {
        this.salary = salary;
    }
}
```

netSalary is calculated data.
→ what happens when we call the setSalary mutator? Is the netSalary field updated?

DATA INTEGRITY WARNING:

- netSalary can contain stale data.
- There is no need to have a netSalary variable; it can always call the netSalary method to get this value.
- We need to re-write calculateNetSalary() to reflect this.

Calculated data

```
public class Employee
{
    private double salary;
    private double deductions;
    :
    public double calculateNetSalary()
    {
        return (salary – deductions);
    }
    public void setSalary(double
salary)
    {
        this.salary = salary;
    }
}
```

netSalary is no longer declared.

calculateNetSalary () now returns the result of the calculation.

→ No calculated data is stored, so no stale data!

Any
Questions?