ICT104

Program Design and Development

Lecture 2– A second look at Classes and Objects

Adopted from: Gaddis & Gaddis (2019) Starting Out with Java: From Control Structures through Objects, 7th Edition.

King's Own Institute

Focus for this week

A Second Look at Classes and Objects

- Overloading Methods and Constructors
- Scope of Instance Fields
- Packages and import Statements
- Static Class Members
- Passing Objects as Arguments to Methods

Activity 1:
Revision Exercise

List any three concepts which you can remember from your previous week class

Overloading Methods and Constructors

- Two or more methods in a class may have the same name as long as their parameter lists are different
- When this occurs, it is called *method overloading*. This also applies to constructors
- Method overloading is important because sometimes you need several different ways to perform the same operation

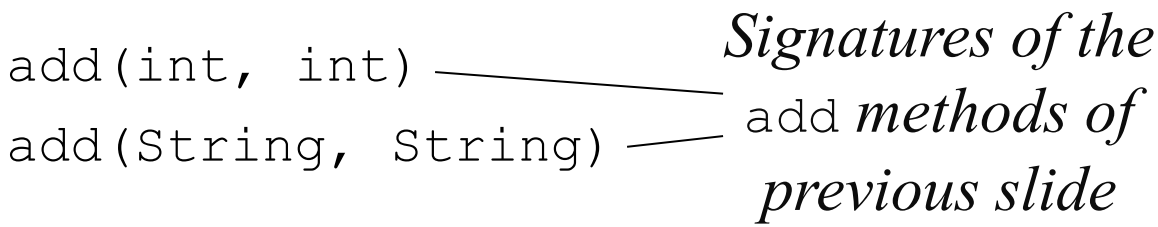
Overloaded Method add

```
public int add(int num1, int num2)
{
  int sum = num1 + num2;
  return sum;
}

public String add (String str1, String
  str2)
{
  String combined = str1 + str2;
  return combined;
}
```

Method Signature and Binding

- A method signature consists of the method's name and the data types of the method's parameters, in the order that they appear. The return type is <u>not</u> part of the signature



- The process of matching a method call with the correct method is known as *binding*. The compiler uses the method signature to determine which version of the overloaded method to bind the call to

`Rectangle` Class Constructor Overload

If we were to add the no-arg constructor we wrote previously to our `Rectangle` class in addition to the original constructor we wrote, what would happen when we execute the following calls?

```
Rectangle box1 = new Rectangle();
Rectangle box2 = new Rectangle(5.0,
  10.0);
```

`Rectangle` Class Constructor Overload

If we were to add the no-arg constructor we wrote previously to our `Rectangle` class in addition to the original constructor we wrote, what would happen when we execute the following calls?

```
Rectangle box1 = new Rectangle();
Rectangle box2 = new Rectangle(5.0,
  10.0);
```

The first call would use the no-arg constructor and `box1` would have a length of 1.0 and width of 1.0

The second call would use the original constructor and `box2` would have a length of 5.0 and a width of 10.0

Activity 2: Poll

When you are working with a _____, you are using a storage location that holds a piece of data.

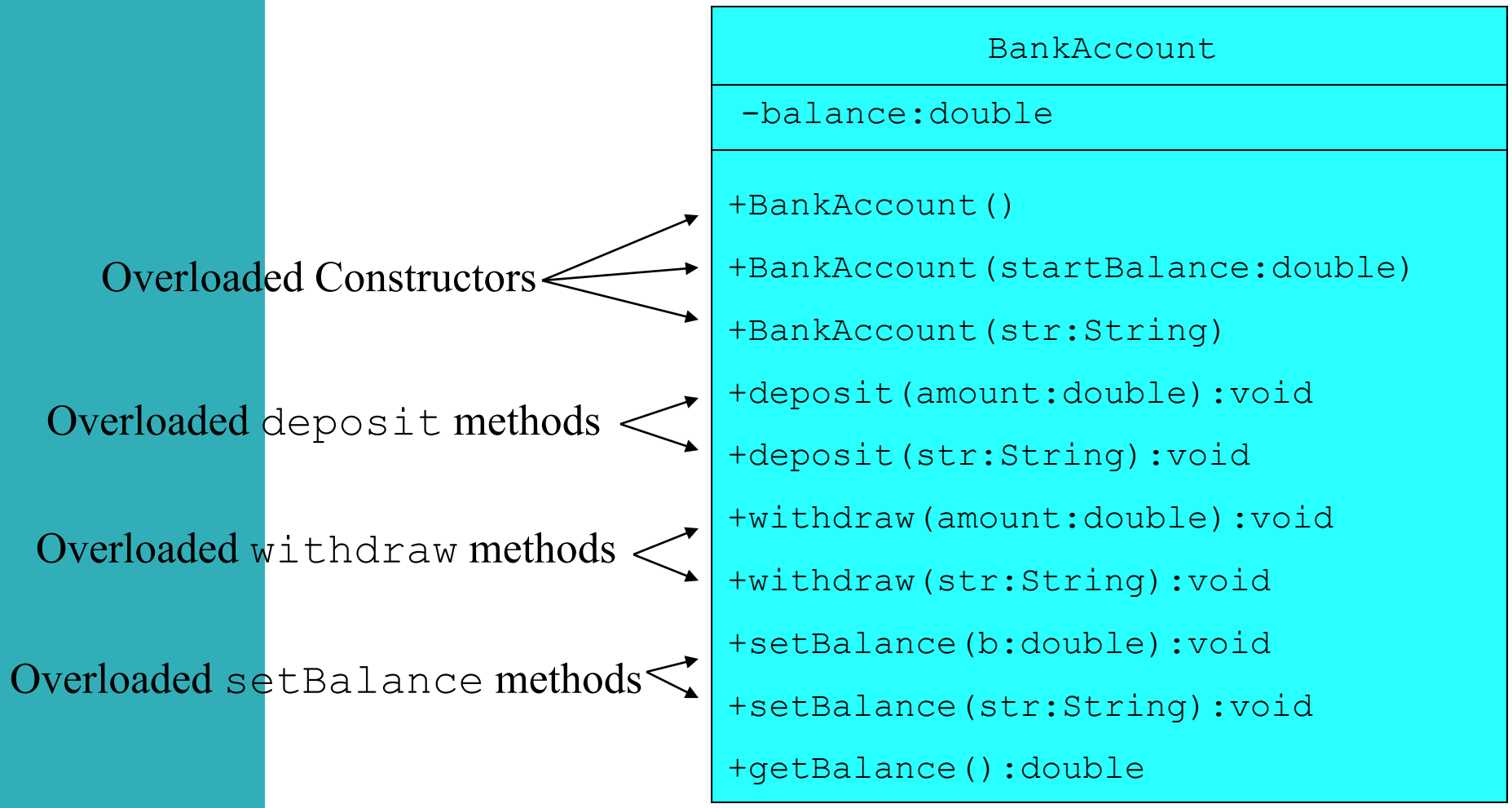
A) primitive variable

B) reference variable

C) numeric literal

D) binary number

The `BankAccount` Example



Scope of Instance Fields

- Variables declared as instance fields in a class can be accessed by any instance method in the same class as the field

- If an instance field is declared with the `public` access specifier, it can also be accessed by code outside the class, as long as an instance of the class exists

Shadowing

- A parameter variable is, in effect, a local variable
- Within a method, variable names must be unique
- A method may have a local variable with the same name as an instance field
- This is called *shadowing*
- The local variable will *hide* the value of the instance field
- Shadowing is discouraged and local variable names should not be the same as instance field names

Packages and `import` Statements

- Classes in the Java API are organized into *packages*

- Explicit and Wildcard `import` statements
  - Explicit imports name a specific class
    - `import java.util.Scanner;`
  - Wildcard imports name a package, followed by an `*`
    - `import java.util.*;`

- The `java.lang` package is automatically made available to any Java class

Some Java Standard Packages

Table 6-2 A few of the standard Java packages

Package	Description
`java.applet`	Provides the classes necessary to create an applet.
`java.awt`	Provides classes for the Abstract Windowing Toolkit. These classes are used in drawing images and creating graphical user interfaces.
`java.io`	Provides classes that perform various types of input and output.
`java.lang`	Provides general classes for the Java language. This package is automatically imported.
`java.net`	Provides classes for network communications.
`java.security`	Provides classes that implement security features.
`java.sql`	Provides classes for accessing databases using structured query language.
`java.text`	Provides various classes for formatting text.
`java.util`	Provides various utility classes.
`javax.swing`	Provides classes for creating graphical user interfaces.

Object Oriented Design
Finding Classes and Their Responsibilities

- Finding the classes
  - Get written description of the problem domain
  - Identify all nouns, each is a potential class
  - Refine list to include only classes relevant to the problem

Object Oriented Design
Finding Classes and Their Responsibilities

- Identify the responsibilities
  - Things a class is responsible for knowing
  - Things a class is responsible for doing
  - Refine list to include only classes relevant to the problem

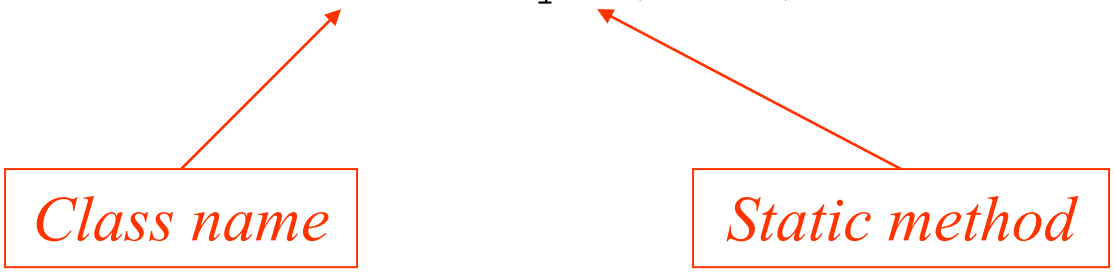
Review of Instance Fields and Methods

- Each instance of a class has its own copy of instance variables
  - Example:
    - The `Rectangle` class defines a `length` and a `width` field
    - Each instance of the `Rectangle` class can have different values stored in its `length` and `width` fields
- Instance methods require that an instance of a class be created in order to be used
- Instance methods typically interact with instance fields or calculate values based on those fields

Static Class Members

- *Static fields* and *static methods* do not belong to a single instance of a class
- To invoke a static method or use a static field, the class name, rather than the instance name, is used
- Example:

```
double val = Math.sqrt(25.0);
```



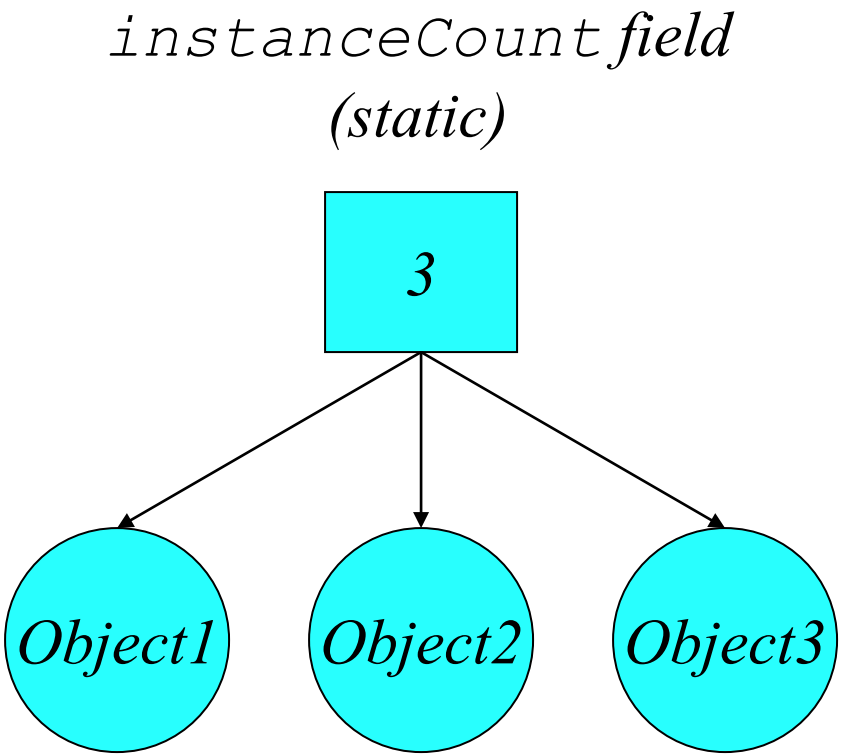
Static Fields

- Class fields are declared using the `static` keyword between the access specifier and the field type

```
private static int instanceCount = 0;
```

- The field is initialized to 0 only once, regardless of the number of times the class is instantiated
  - Primitive static fields are initialized to 0 if no initialization is performed

Static Fields



Static Methods

- Methods can also be declared static by placing the `static` keyword between the access modifier and the return type of the method

```
public static double
  milesToKilometers(double miles)
{…}
```

- When a class contains a static method, it is not necessary to create an instance of the class in order to use the method

```
double kilosPerMile =
  Metric.milesToKilometers(1.0);
```

- Examples: Metric.java, MetricDemo.java

Static Methods

- Static methods are convenient because they may be called at the class level
- They are typically used to create utility classes, such as the `Math` class in the Java Standard Library
- Static methods may not communicate with instance fields, only static fields

Activity 3: Poll

Instance methods do not have this key word in their headers:

A) public

B) static

C) private

D) protected

Activity 4: Poll

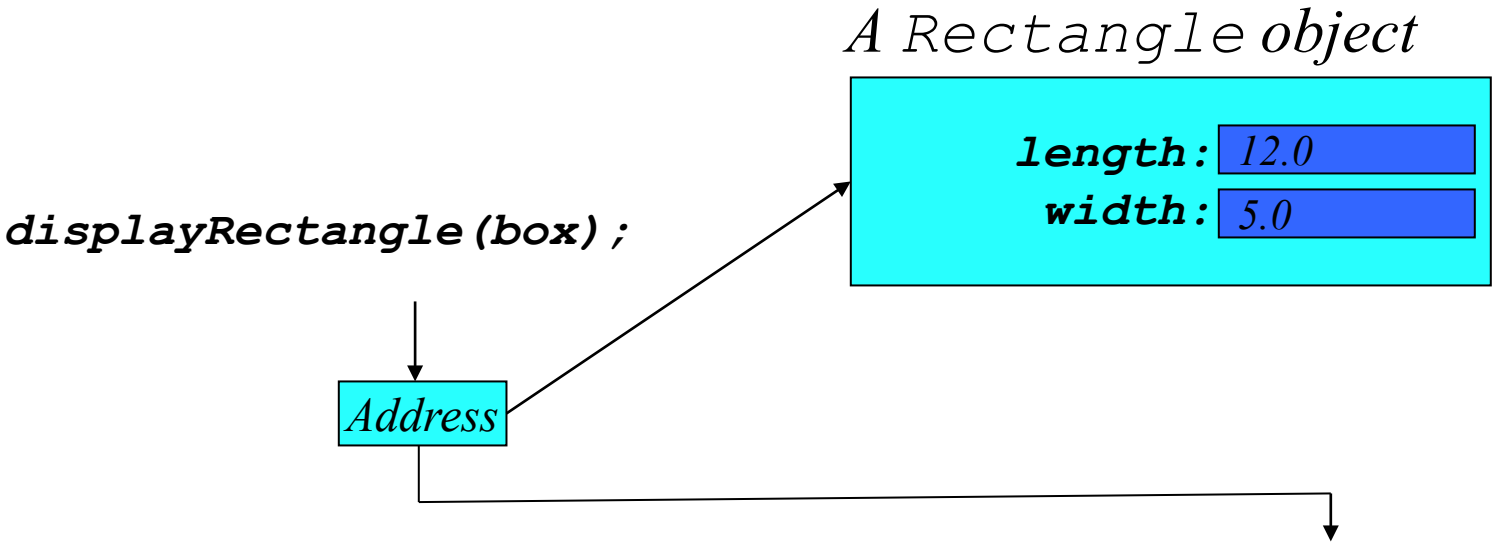
State TRUE or FALSE:

The term "no-arg constructor" is applied to any constructor that does not accept arguments

Passing Objects as Arguments

- Objects can be passed to methods as arguments
- Java passes all arguments *by value*
- When an object is passed as an argument, the value of the reference variable is passed
- The value of the reference variable is an address or reference to the object in memory
- A *copy* of the object is *not passed*, just a pointer to the object
- When a method receives a reference variable as an argument, it is possible for the method to modify the contents of the object referenced by the variable

Passing Objects as Arguments




```
public static void displayRectangle(Rectangle r)
{
  // Display the length and width.
  System.out.println("Length: " + r.getLength() +
                     " Width: " + r.getWidth());
}
```

Summary of todays lesson

Objects and Classes

- Designing classes for the purpose of instantiating objects
- Class fields, methods and UML diagrams
- Identifying classes and their responsibilities within a problem domain
- Class Constructor

Activity 5:
Reflection Exercise

List any four concepts you have learnt in today's lesson

Activity 6: Homework Exercise

Write a program using Eclipse or NetBeans to implement any one concept you have learnt in today's lesson