# *Programming II (420-B20-HR)*

# *Lab 3 – Inheritance and Polymorphism*

Date assigned: Wednesday, February 3, 2016

Date due: **Wednesday, February 3, 2016**

**Learning Objectives**

At the end of this lab, the student will be able to:

1. create a subclass of an existing class;
2. describe the order of constructor execution in an inheritance hierarchy;
3. call a superclass constructor;
4. use protected visibility to make instance variables accessible in subclasses;
5. override a superclass method;
6. declare an object as a superclass object and instantiate it as a subclass object;
7. explain how polymorphism works in method calls;
8. use the cast operator to call a subclass method from an object declared in the superclass;
9. use the **getClass()** method to determine the class of an object;
10. declare and use abstract classes and methods.

**Methods Used:**

*Call the superclass constructors:*

super()

super(*parameter list*)

*Call superclass methods:*

super.*methodName*(*parameter list*)

*Determine the class of an object:*

*objectName*.getClass()

**To Be Handed In:**

1. The ***username\_*B20\_L03\_Project** folder should zipped and be uploaded to **Moodle**.

**To Start:**

1. Download and unzip the folder **B20\_L03\_Project** from the **Moodle**. Rename it to ***username\_B20\_L03\_Project***.
2. Start **Eclipse** and select your **420-B20\Labs** folder as your workspace.
3. Create a new **Java Project** called ***username\_B20\_L03\_Project***.

# Inheritance

***Purpose***: Learn to use a subclass to inherit attributes and methods.

***To Do***:

## Open the **Product** class. It contains most of the attributes of movies and games. But it does not contain the director for a movie or the platform for a Game. Instead of creating two new classes for games and movies that contain all the attributes and methods of each, we will create subclasses of the **Product** class. The **Movie** subclass will inherit all the attributes and methods from the **Product** class and add a **director** attribute. The **Game** subclass will inherit all the attributes and methods from the **Product** class and add a **platformCode** attribute.

## Create a new class called **Movie**.

## Add **extends** **Product** to the **Movie** class header.

## Add a private **String** instance variable called **director**.

## Add an accessor and mutator for the **director** instance variable.

## Create a default constructor. It should set **director** to "*Unknown*". Add the following statement at the end of the constructor:

**System.out.println("In Movie constructor");**

## Create a second subclass of **Product** called **Game**.

## Add a private integer instance variable called **platformCode** to **Game**. Add an accessor and mutator for it.

## Create the default **Game** constructor to set **platformCode** to **0**. Add the following statement at the end of the constructor:

**System.out.println("In Game constructor");**

# Constructors in an Inheritance Hierarchy

***Purpose:*** Learn the order in which constructors execute when using classes with inheritance.

***To Do:***

## Create a new class called **InheritanceTest**. It should contain a **main()** method which instantiates a **Product** object called **product**, a **Movie** object called **movie** and a **Game** object called **game**. Print a message before instantiating each object describing the object being instantiated. Run the program.

Fill in the following table after running the program:

| **Class** | **Constructor order** |
| --- | --- |
| Product | In Product() constructors |
| Movie | In Product() constructors  In Movie constructor |
| Game | In Product() constructors  In Game constructor |

# Overloaded Constructors

***Purpose:*** Learn to call superclass constructors from subclass constructors.

***To Do:***

## Add the following constructor to the **Movie** class to instantiate objects with a title.

**public Movie(String t)**

**{**

**director = "Unknown";**

**System.out.println("In Movie(String) constructor");**

**}**

## Since **title** is an inherited instance variable of the superclass **Product** and since the **Product(String)** constructor initializes the title, we should simply call the **Product(String)** constructor to perform the initialization. To do this, add a call to the superclass constructor at the beginning of the **Movie** constructor:

**super(t);**

## Modify the **movie** instantiation in **InheritanceTest** to use the new constructor:

**Movie movie = new Movie("Star Wars: The Force Awakens");**

## Add the following statement after the instantiation to see if the title was properly initialized:

**System.out.println("The title for movie is "**

**+ movie.getTitle());**

## Run **InheritanceTest** to test the new constructor. What is the order of the constructor calls for **movie**?

In Product (String) constructor\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

In Movie(String) constructor\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The title for movie is Star Wars: The Force Awakens\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Change the order of the statements in the **Movie(String)** constructor so that the director assignment statement is first. What happens? \_\_Error. Super(t); isn’t the first thing in the constructor.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Rule**: If a constructor includes a call to its superclass constructor, that call must be the **first** statement in the subclass constructor.

## Correct the order of the statements in **Movie(String)**.

## Add the following constructors to **Game** and **Movie**. In each constructor, include a call to the corresponding superclass constructor, initialize the director/platformCode to the default if necessary and add a println() statement with the constructor signature:

**Movie(String t, int year, String c, String f)**

**Game(String t)**

**Game(String t, int year, String c, String f)**

**Movie(String t, int year, String c, String f, String d)**

-- after calling the superclass constructor, initialize **director** to **d**

**Game(String t, int year, String c, String f, int p)**

-- after calling the superclass constructor, initialize **platformCode** to p

## The **productNumber** in the **Movie** class is to be prefixed with "M" and the **productNumber** in the **Game** class is to be prefixed with "G". Add a **private** method called **setProductNumber()** to the **Movie** class. It should contain the following statement:

**productNumber = "M" + productNumber;**

What happens? \_\_ Multiple markers at this line\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

- The field Product.productNumber is \_\_\_\_\_\_\_\_\_\_\_\_\_\_

not visible\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

- The field Product.productNumber is \_\_\_\_\_\_\_\_\_\_\_\_\_\_

not visible\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Explanation: productNumber** is **private** which means that only the **Product** class can access it. To make it accessible from a subclass, it must be made **protected**.

## Change all the private instance variables in the **Product** class to be **protected**.

## Add a call to **setProductNumber()** to all the constructors in the **Movie** class.

## Add a private **setProductNumber()** method to prefix **productNumber** with "G" to the **Game** class. Add a call to the method to all the constructors in the **Game** class.

## Add statements in **InheritanceTest** to display the productNumbers of each of the objects.

## Run **InheritanceTest** to test your changes. What is the first product number? \_G2\_\_\_\_\_\_\_\_

## Open **productNumber.txt**. What is the value of the next product number? 1003\_

We want to use this number for the first product to be added. The **open()** method of the **Product** class reads this file and sets **nextProductNumber** appropriately.

## Add a call to **Product.open()** at the beginning of the **main()** method in **InheritanceTest**. Add a call to **Product.close()** at the end of the **main()** method.

## Run **InheritanceTest** again. What is the first product number now? G1005\_\_\_\_\_

## We want to prevent a user from creating a product without first initializing **nextProductNumber**. To do this we will add a static Boolean variable that is true if the **nextProductNumber** has been initialized and false if it has not. In the **Product** class:

### Add a static Boolean private instance variable called **nextNumberIsInitialized**. Initialize it to **false**.

### Set **nextNumberIsInitialized** to **true** at the end of the **initializeNextNumber()** method.

### Add an if statement to the **setProductNumber()** method (not to the **setProductNumber(String)** method) to test **nextNumberIsInitialized**. If it is true, set the product number and update **nextNumber**. If it is false, issue the following error message: "*ERROR: next number is not initialized. Call Product.open() before instantiating an object*" and exit with a code of -3.

## Comment out the **Product.open()** call in **InheritanceTest** to test your changes. The program should exit when initializing the product.

## Uncomment the **Product.open()** call and run it again. It should work this time.

# Method Overriding and Polymorphism

***Purpose:*** Learn to override a superclass method in a subclass.

***To Do:***

The format code and the category code for movies and games have different possible values. For a movie the possible values are:

| **Movie Category Code** | **Meaning** |  | **Movie Format Code** | **Meaning** |
| --- | --- | --- | --- | --- |
| 0 | Unknown (default) |  | 1 | DVD (default) |
| 1 | Family |  | 2 | DVD-HD |
| 2 | Action |  | 3 | Blu-Ray |
| 3 | Comedy |  |  |  |
| 4 | Western |  |  |  |
| 5 | Drama |  |  |  |
| 6 | Horror |  |  |  |
| 7 | Sci-Fi |  |  |  |

For a game, the possibilities are:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Game Category Code** | **Meaning** |  | **Game**  **Format**  **Code** | **Meaning** |
| 0 | Unknown (default) |  | 1 | DVD |
| 1 | RPG |  | 2 | CD |
| 2 | Action |  |  |  |
| 3 | Education |  |  |  |

The **Product** class has two stub methods to determine the codes from the String input: **setCategoryCode(String)** and **setFormatCode(String)**. We now want to **override** these methods in the subclasses to properly set the code values.

**Method Overriding**: redefining the implementation of a method that a class inherits from its superclass.

## Change the instantiation of **movie** in **InheritanceTest** to:

**Movie movie =**

**new Movie("Star Wars: The Force Awakens", 2015, "Family", "DVD");**

and add a statement to display the category:

**System.out.println("The category code is " +**

**movie.getCategoryCode());**

## Run **InheritanceTest**.

What is the category code? \_Category code is 0\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Where is it set? \_\_setCategoryCode method\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Put 1, 2 or 3 on the blank in front of each of the following methods where the number represents the order in which the method is executed:

\_3\_\_ Movie(String, int, String, String)

\_1\_\_ Product(String, int, String, String)

\_2\_\_ setCategoryCode(String) in Product

## Add the following method to the **Movie** class:

**public void setCategoryCode(String categoryName)**

**{**

**System.out.println("In the Movie setCategoryCode(String)"**

**+ " method");**

**if (categoryName.equalsIgnoreCase("Family"))**

**categoryCode = 1;**

**else if (categoryName.equalsIgnoreCase("Action"))**

**categoryCode = 2;**

**else if (categoryName.equalsIgnoreCase("Comedy"))**

**categoryCode = 3;**

**else if (categoryName.equalsIgnoreCase("Western"))**

**categoryCode = 4;**

**else if (categoryName.equalsIgnoreCase("Drama"))**

**categoryCode = 5;**

**else if (categoryName.equalsIgnoreCase("Horror"))**

**categoryCode = 6;**

**else if (categoryName.equalsIgnoreCase("Sci-fi"))**

**categoryCode = 7;**

**else**

**categoryCode = 0;**

**} // setCategory()**

## Run **InheritanceTest** again.

What is the category code now? \_\_1\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Put the order number beside each of the following methods:

\_4\_\_ Movie(String, int, String, String)

\_2\_\_ Product(String, int, String, String)

\_1\_\_ setCategoryCode(String) in Product

\_3\_\_ setCategoryCode(String) in Movie

Which version (**Movie** or **Product**) of **setCategoryCode()** executes? Movie\_\_\_

Why? \_Because the Movie one is the one that gives it a code and not a default \_

**Polymorphism**: a dynamic technique in which the method to be performed as the result of a call to an overridden method is determined by the run-time class of the object executing the method.

## Add a **setFormatCode(String)** method to the **Movie** class. Set **formatCode** according to the table at the beginning of this section. If the format name is invalid set **formatCode** to 0.

## Add a **setCategoryCode(String)** method to the **Game** class. Set **categoryCode** according to the table at the beginning of this section. If the category name is invalid, set the **categoryCode** to 0. Add a **println()** that states it is executing the **Game setCategoryCode()** method.

## Add a **setFormatCode(String)** method to the **Game** class. Set **formatCode** according to the table at the beginning of this section. If the format name is invalid set **formatCode** to 0.

## Change the instantiation of **game** in **InheritanceTest** to:

**Game game =**

**new Game("Rock Band 2", 2015, "Education", "DVD");**

## Add **println()** statements to print the title and the category code.

## Run **InheritanceTest** to test your changes.

# An object can be of more than one type

***Purpose:*** Learn that an object can be more than one type.

***To Do:***

## Create a new class called **InheritanceTest2**. It should contain a **main()** method.

## In the **main()** method of **InheritanceTest2**:

### declare a **Product** object called **product**. Initialize it to null. (Do not instantiate it.)

### add a **Scanner** object called **input** to read from the keyboard

### add the following code:

Product.*open*();

System.*out*.print("Enter product type (M - Movie, G - Game) > ");

**char** productType = Character.*toUpperCase*(input.nextLine().charAt(0));

System.*out*.print("Enter title> ");

String title = input.nextLine();

System.*out*.print("Enter category> ");

String category = input.nextLine();

**if** (productType == 'M')

product = **new** Movie(title, 2015, category, "DVD");

**else**

**if** (productType == 'G')

product = **new** Game(title, 2015, category, "DVD");

**else**

product = **new** Product(title, 2015, category, "DVD");

Product.*close*();

### add the following statement to display the class for the **product** object:

**System.out.println("The class for product is "**

**+ product.getClass());**

## Run the program three times. Try creating a movie, a game and a product.

What does **getClass()** return? \_\_Nothing \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Which version of **setCategoryCode(String)** gets executed for the product?

\_In the Product setCategoryCode(String) method\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Change **product** to a **Movie** object. Try compiling it.

What happens? \_\_Type Mismatch error\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Change **product** back to a **Product** object.

# Use the cast operator

***Purpose:*** Learn to use the cast operator to call a subclass method.

***To Do:***

## We want to input the director name for a movie. In **InheritanceTest2**, modify the **if** statement for a Movie as follows:

**if (productType == 'M')**

**{**

**product = new Movie(title, 2015, category, "DVD");**

**System.out.print("What is the director's name? ");**

**String director = input.nextLine();**

**product.setDirector(director);**

**System.out.println("The director of the movie is "**

**+ product.getDirector());**

}

## Compile **InheritanceTest2**. What happens? \_Argument not applicable \_\_\_\_\_\_

**Explanation**: Because **product** is declared as a **Product**, only **Product** methods are available to it.

## In order to use a method of the instantiated subclass, you must first cast the object to the subclass. Change the call to **setDirector()** to:

**((Movie)product).setDirector(director);**

## Correct the call to **getDirector()**.

## Run **InheritanceTest2** to test your changes.

## Add a **setPlatformCode(String)** method to the **Game** class. It should set the platform code to the appropriate value. (See the **setCategory(String)** method to see how to do this.) The platform codes are shown in the following table:

| **Platform Code** | **Meaning** |
| --- | --- |
| 0 | Unknown |
| 1 | Sony PlayStation |
| 2 | Microsoft xBox |
| 3 | Nintendo Wii U |
| 4 | Nintendo Wii |

## In **InheritanceTest2**, modify the **if** statement for a game to prompt for and read the platform. Call the **setPlatformCode()** method you just wrote to set the platform code. Display the platform code.

## Run **InheritanceTest2** to test your changes.

**Don't forget to format all your classes before zipping and submitting the project.**

# Homework

## Complete the **Week 3 Quiz** on Moodle by Feb. 7.