Introduction to Java

CS9053

Prof. Dean Christakos

MIDTERM

Thursday, July 6th, 2023

DUE: Thursday, July 14th, 2023

NO GRACE PERIOD OR EXTENSION

**Part I: Written Questions**

1. Unlike C++, Java does not have “pointers.” How, then, are things passed by reference? What are data types that Java maintains references to?

Reference data types are all non-primitive data types such as strings, arrays, classes, interfaces. It refers to the memory locations of dynamically created objects.

1. What is the difference in purpose between the equality operator (==) and the equals method?

(==) checks if both objects point to the same memory location and equals() checks if the values and type of the different objects are the same.

1. Why do you have to write a main method in a class MyClass to run a Java program via “java MyClass” ? What’s going on here when the program runs?

The main method is how the Java Virtual Machine (JVM) starts a class without instantiating the class. It needs to be public so that the JVM can find it, static so that it can be loaded into memory, and return type of void so it doesn’t need to return anything. It also has a string array for initial arguments, which can or cannot be used. Java program works by taking the source code and compiling it to check for syntax errors and generates the binary code which is then executed by the JVM.

For questions 4, 5, and 6:

public class Fruit {

String name;

Fruit() {

this.name = “unknown fruit”;

}

Fruit (String name) {

this.name = name;

}

public String getName() {

return this.name;

}

public void setName (String name) {

this.name = name;

}

public void toString() {

return “Fruit [name=” + this.name + “]”;

}

}

public class Apple extends Fruit {

Apple() {

System.out.println(“the name of the fruit is “ + this.getName());

this.setName(“Apple”);

}

public void toString() {

return “Apple [“ + super.toString() + “]”;

}

}

Ignore any syntax errors or wrongly capitalized key words. Assume the code compiles and does “the right thing”.

Let’s say I create two objects:

Fruit f1 = new Fruit();

Fruit f2 = new Apple();

1. If you create an Apple object, the console will print out “the name of the fruit is unknown fruit”. Why will this value for the name have been set?

The name isn’t being set to apple until after that line is printed out, so it would be setting the name that is made by default from the super class.

1. Why can I assign an Apple object to a Fruit reference variable f2 without any errors but Apple a = f2; will give a compile error?

Because apple is a subclass of Fruit, so it is also a fruit but a fruit isn’t always an apple, it can be just a fruit or any other subclass such as banana, orange, etc. and those other subclasses wouldn’t fall under apple.

1. If you execute f2.toString(), the output will be “Apple [Fruit [name=Apple]]”. Why is this the output, which has executed the toString() of the Apple class rather than the toString() of the Fruit class, since f2 is a Fruit reference variable?

The method in the subclass always overrides the method in the superclass of the same name.

1. Explain what the this keyword means and where it is used (give at least two examples).

The keyword “this” refers to the current object in the class. It’s meant to help differentiate between class attributes and parameters of the same name.

Example 1:

public class Vehicle {

private int wheels;

private double space;

private String color;

private int id;

private static int next\_id = 0;

public Vehicle(int wheels, double space, String color) {

this.wheels = wheels;

this.space = space;

this.color = color;

next\_id = next\_id + 1;

this.id = next\_id;

}

}

In the above example “this” is used to reference the class attributes and assign values from the constructor’s parameters to the class attribute values of the same name.

Example 2:

public String toString() {

String s = "This vehicle has " + this.getWheels() + " wheel(s), " + this.getSpace() + " cubic feet of cargo space, is "

+ this.getColor() + " and has ID number " + this.getiD() + ".";

return s;

}

In this example, “this” is being used to call methods of the current class object in another method from the class.

1. What is the difference between a Checked Exception and an Unchecked Exception? If you create your own Exception subclass as a subclass of Exception, it is Checked. How would you instead make it an Unchecked Exception?

A checked exception is an error that can happen from outside factors of the program., something that is predictable and can be checked at compile time, will be required to have a throw clause or try-catch block used to deal with the exception. An unchecked exception has to do with programming logic, it isn’t checked at compile time and isn’t required to be caught or declared in a throw clause. To create an unchecked exception, we would have to create an Exception subclass as an extension of RuntimeException.

1. GUI components have an “addActionListener” method which takes as an argument an object that implements the ActionListener interface. However, we don’t actually have to create a class that implements the ActionListener interface, instantiate it as an object, and pass it to addActionListener. Instead we can just pass in a lambda function. Why is this allowed and how does it know to call that lambda method?

The ActionListener interface only has one abstract method actionPerformed, so we can use a lambda expression instead, since that is just the implementation of a functional interface and ActionListener is a functional interface because it only has one abstract method, and because of that it will assume the lambda expression is that one abstract method.

1. Explain the difference between procedural programming and event driven programming. Use GUIs as an example that demonstrates the difference.

The difference is in program flow, in procedural programming the execution is in a given order and runs from beginning to end. In event-driven programming, the execution is determined by events the system is monitoring for, it has no set order of execution. In a calculator GUI, the system is responding to different inputs from the user, if the user decides to change a number or operation, the system must respond to that event and update the output. In procedural programming, it’s not on the lookout for changes in the numbers or operation, it just runs straight forward from beginning to end.

**Part II: Coding Section**

1. Inheritance

Demonstrate your ability to program using inheritance in the Java programming language.

Demonstrate your knowledge of proper equals implementations in the Java programming language.

Instructions

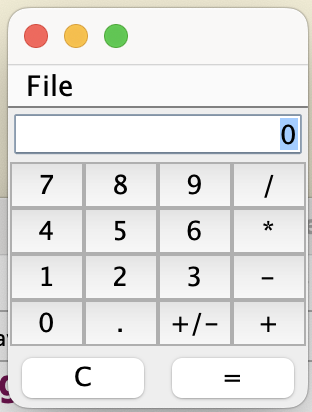
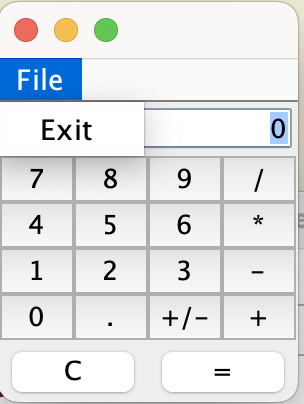
* There are three tasks to complete

Create an object hierarchy which mimics the classification for university affiliates. This should proceed logically.

* The hierarchy should be contained within package edu.nyu.cs9053.midterm.hierarchy
* There should be an **abstract** base class called UniversityAffiliate
  + There should be the following subtypes. Subtypes in bold should be abstract:
    - **Student**
    - UndergraduateStudent
    - GraduateStudent
    - **Employee**
    - **Faculty**
    - Staff
    - Administrator
    - Lecturer
    - AssistantProfessor
    - AssociateProfessor
    - Professor
  + Each of the subtypes should have the following methods:
    - getName returns a name for the person as a String.
    - getAge returns the age of the person as an int.
  + Students should have the method:
    - isMatriculated which returns a Boolean
  + Employees should have the methods:
    - getSalary which returns a salary as a double
  + Faculty should have the methods:
    - isTenured which returns a boolean
    - isAdjunct which returns a boolean
  + Staff and Administrator classes should have the method getTitle which returns a String indicating their job title
* There should be an equals() method to show if two faculty objects have the same values (including name)

Note: for those who need some hints about how university employees work, faculty are anyone who teach. Administrators are generally not considered faculty.

1. GUIs and Events

Create a calculator. As you can see, it has buttons and a text field where numbers and results are entered. “=” or the return button in the text field should calculate the result.

You should create the GUI, and it should have a File menu with a menu item “Exit” that exits the program, which should also terminate if you close the window.

Hint: the text field in in the NORTH section of the layout, and the “=” and “clear” button are in a panel in the SOUTH section of the layout.

The clear button should clear the last entered number, and after doing so, the label on the button should change to “AC” such that if you press it again, all the numbers should reset. You should have two numbers kept track of—each term of an equation. The result should become the first term of the equation, and the next number entered should be the second term of whatever equation you put in.

You’ll have to figure out what the optimal size is to get the buttons to line up correctly.

It doesn’t have to be THAT sophisticated. Just a basic calculator.