Rochester Institute of Technology
National Technical Institute for the Deaf
Information and Computing Studies Department

Name: Edward Riley

NACA.161 Programming Fundamentals II
In-class Exercise #20 – Interfaces

Overview

This exercise is designed to review interfaces. In particular, this exercise will require you to inherit from concrete classes and interfaces.

Create an Interface

1) Create an interface called **MyInterface**. Save and compile the file.

2) Add the following attribute exactly as shown:

   private int var;

3) Save and compile the file.

   Did it compile? No

   Why not?

   access modifier private is unallowed

   What access modifiers must be used for all attributes in an interface?

   public

4) Remove this variable and add a constant called INTER_CONST with a value of 10.0

5) Add the following method exactly as typed:

```
public void myInterMethod()
{
    System.out.println("Hello");
}
```

Did it compile? No

Why not?

Because you cannot have a body

6) Fix **myInterMethod** so that it compiles successfully.

Create a Concrete Class

7) Create a class called **MyConcrete**. Save and compile the file.

8) Add the following attribute:

Name	Type	Range of values
conVar	int	Between 10 and 20

9) Create an accessor for **conVar**. Save and compile the file.

10) Create a mutator for **conVar**.
    a) Make sure you only allow the range of values listed in the table.
    b) If the passed-in value is out of range, print an error message that states the concrete value was invalid and displays the invalid value.
    c) Save and compile the file.

11) Create a 1-argument constructor that
    d) Uses the mutator to set **conVar** to the value passed as the parameter.
    e) Save and compile the file.

12) Create a default constructor that calls the 1-argument constructor with a value of 15.0. Save and compile the file.

What code did you use to call the 1-argument constructor?

public MyConcrete(int _conVar)

13) Add the following method:

```
public void myConMethod()
{
  System.out.println("I am concrete");
}
```

Make sure this class compiles before continuing.

Inherited Class

14) Create a class called **MyInheritedClass** that inherits everything from **MyInterface**. NOTE: Do not add any code, merely create the header and a pair of curly braces.

What keyword did you use to inherit from **MyInterface**?

implements

15) Attempt to compile **MyInheritedClass**.

Why did it not compile?

My Concrete is not abstract - doesnot override myInterMethod

16) Correct the error by defining **myInterMethod**. Make this method print the message: "I am myInterMethod"

Make sure this class compiles before continuing.

Additional Inheritance

17) Now also make **MyInheritedClass** also inherit everything from **MyConcrete**.

How did you define the class header to make this class inherit from both **MyInterface** and **MyConcrete**?

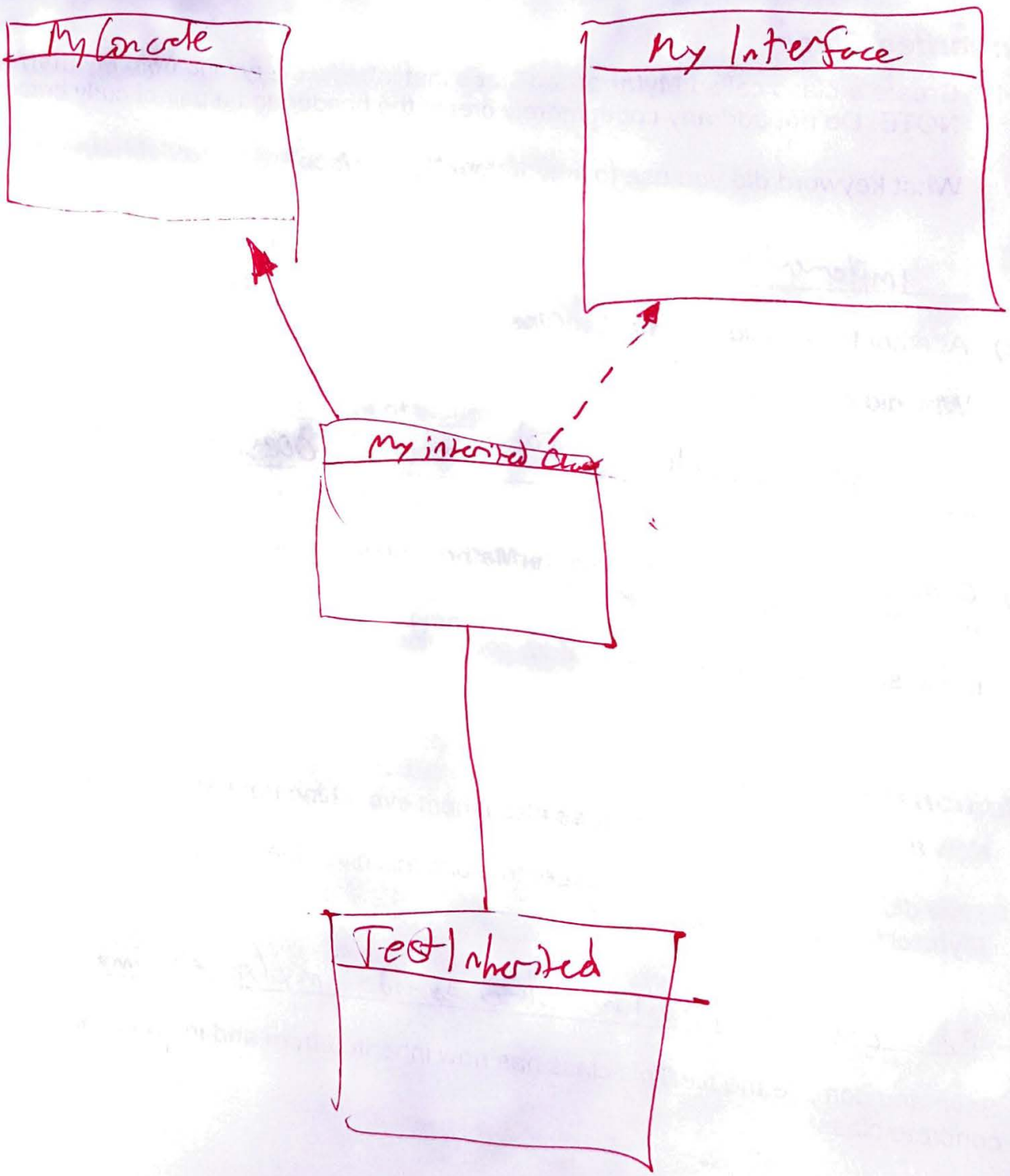
public class MyInheritedClass extends myConcrete implements MyInterface

Save and compile the file.This class has now inherited from and interface and a concrete class!

18) Add a 1-argument constructor that passes the parameter to its superclass. Save and compile the file.

19) Add a default constructor that passes a value of 10 to the 1-argument constructor. Save and compile the file.

20) Draw the UML diagram of the classes that you have created to this point.



Test Class

Now lets use our newly created classes. Even if you know certain steps are incorrect, do them anyway and answer the questions.

21) Create a class called **TestInheritance** that includes a main method but no code. Save and compile.

22) Create a **MyInterface** object. Save and compile the file.

Did it compile? No

Why or why not?

It was not abstract

If program did not compile, delete the object so that the program compiles.

23) Create a **MyConcrete** object. Save and compile the file.

Did it compile? Yes

Why or why not?

Instantiable

If the program did not compile, delete the object so that the program compiles.

24) Create a **MyInheritedClass** object. Save and compile the file.

Did it compile? Yes

Why or why not?

Instantiable

Use **MyInheritedClass** objects for the rest of this practice exercise.

25) Using the **MyInheritedClass** object, call the **myInterMethod** method. Compile and run the **TestInheritance class.**

Did it work? Yes

What did it print out?

I am my InterMethod

Where was the method declared?

MyInterface

Where was the method defined?

MyConcrete

What is the difference between declaring a method and defining a method?

Declaring is using method signatures as defining is re-altering in method body

26) Now add code to the main method to call the **myConMethod**. Save and compile the file.

What did it display?

I am Concrete

Where was this method defined?

My Concrete

Notice that you are calling several methods on the same object but they are defined in different classes.

27) Add code to display the current value of conVar. Save and compile the file.

What method did you call?

getConVar

In which class was this method defined?

My Concrete

Now run the program. Correct any errors so that the program executes correctly.

What is its value of conVar? 10.0

28) Add code to create another **MyInheritedClass** object using the 1-argument constructor and pass in a value of 30. Save, compile and run the file.

Did it generate an error message? Yes

If not, correct the problem.

Where did the error message come from? MyConcrete

29) Add code to display the value of **conVar** from the object in the previous step. Save, compile and run the file.

What is the value? 0

Why is it not equal to 30?

Because we changed the program

30) Add code to create another **MyInheritedClass** object using the 1-argument constructor and pass in a value of 12. Save, compile and run the file.

31) Add code to display the value of **conVar** from the object in the previous step. Save, compile and run the file.

Is it equal to 12? Yes

If not, correct the problem

Signoff

When you complete all of the steps successfully and answer all of the questions, contact your instructor to check if your application(s) executes correctly and to review your code. We will initial the line below.

_____**Successful execution of code**

If you do not finish the program during the class period, contact your instructor to check to review your code and initial below.

_____**Code not completed during lab time**

You may then submit your work at the start of next class. You may not use the work period of the next class to complete this assignment. If you do not have a signature, then you cannot receive any points for this assignment.