***CSC 3020***

***Java Programming***

**Assignment 07**

**40 points**

**Due 11/27/2023 (11:45 A.M.)**

Assignment Objectives:

* To design and use abstract classes.
* To specify common behavior for objects using interfaces.
* To define interfaces and define classes that implement interfaces.
* To define a natural order using the **Comparable** interface.
* To explore the similarities and differences among concrete classes, abstract classes, and interfaces.

**Solution to this assignment will not be posted on Canvas; however, any question can be discussed in the class upon request of a student.**

All assignments must be submitted by the Canvas. **No email or hard copy** is accepted. You must follow the following format:

1. For non-programming questions, use a word file to type your answers. Don’t use the text box on the Canvas to answer the questions or to write comments, we will not read it.
2. State your answer clearly.
3. For programming questions, include only the source file for each problem.
4. Submit your file to the Canvas. You must submit your assignment on time; otherwise, you will receive zero. In addition, you cannot submit your file more than one time.
5. There will be several folders on the Canvas. You need to upload your file(s) using the correct folder on the Canvas.
6. Name each file: “Assignment Number(Question number(s))”.
7. To upload your file(s):

* In Course Navigation, click the Assignments link.
* Click the title of the assignment.
* Click the **Submit** Assignment button.
* Add **File**. ...
* Add Another **File**. ...
* **Submit** Assignment. ...
* View **Submission**.

**It is your responsibility to make sure that each file is uploaded correctly. If you uploaded a wrong file, you receive zero; files will not be accepted after due date even if you have a prove that the file is created before the due date.**

**Make sure you review the Cheating & Plagiarism policy on Canvas.**

Answer questions 1 to 4 in a separate single .txt file; include only your answers with questions numbers. Write a program for each of Q.5 - Q.6; save each program in a .txt file. Submit total of 3 .txt files by the due date.

**Question 01 (9 points)**

True or false?

1. An abstract class can be used just like a nonabstract class except that you cannot use the new operator to create an instance from the abstract class.   
   True
2. An abstract class can be extended.

TRUE

1. A subclass of a nonabstract superclass cannot be abstract.   
   FALSE
2. An abstract method must be nonstatic.  
   TRUE
3. An interface is compiled into a separate bytecode file.   
   TRUE
4. An interface can have static methods.   
   TRUE
5. An interface can extend one or more interfaces.   
   TRUE
6. An interface can extend an abstract class.   
   FALSE
7. An interface can have default methods.  
   TRUE

**Question 02 (4 points)**

Answer the following questions?

1. Suppose A is an interface. Can you create an instance using new A()?  
   No. You cannot create instance of interfaces as objects, they are created by using the classes that implement them.
2. Suppose A is an interface. Can you declare a reference variable x with type A like this?

A x;  
//Yes.

1. Show the error in the following code:

**interface** A {

**void** m1();

}

**class** B **implements** A {

**void** m1() {

System.out.println("m1");

}

}  
//Class B needs the public access modifier to make the interface be implemented.

1. You can define the compareTo method in a class without implementing the Comparable interface. What are the benefits of implementing the Comparable interface?  
   The point of an interface is to in part make sure that all code is standardized, using the comparable interface achieves this standardization which forces compatibility.

**Question 03 (4 points)**

Which of the following is a correct interface?

(a)

**interface** A {

**void** print() { }

}

(b)

**interface** A {

**void** print();

}

//B is a correct interface. The other ones include brackets for their methods, which is a no no.

(c)

**interface** A {

**default** **void** print() {

}

}

(d)

**interface** A {

**static** **int** get() {

**return** 0;

}

}

**Question 04 (4 points)**

The following questions are based on the Edible interface and the classes defined in Ch. 13. For each question, answer if the code can compile, can run. If not, give a reason. If it runs, give the output.

a. Edible x = **new** Tiger();   
//Does not compile. Error: Tiger Cannot be converted to Edible because it is not a compatible types. (

b. Edible x = **new** Chicken();

System.out.println(x.sound());   
//Does not compile as it cannot find the symbol for x

c. Edible x = **new** Chicken();

System.out.println((Animal)x.sound());   
//Does not compile as it cannot find the symbol for x.

d. Edible x = **new** Chicken();

System.out.println(((Animal)x).sound());  
//COMPILES:  
//Chicken: Cock-a-doodle-doo

**Question 05 (9 points)**

Design an interface named **Colorable** with a **void** method named **howToColor()**. Every class of a colorable object must implementthe **Colorable** interface. Design a class named **Square** that extends **GeometricObject** and implements **Colorable**. Override method **howToColor** todisplay the message **Color all four sides**. The **Square** class contains a **private** data field **side** with getter and setter methods, and a constructor for constructinga **Square** with a specified side.

Class **GeometricObject** can be downloaded from Assignment 7 link on Canvas.

Test your code by creating an array of GeometricObject and initialize it with three instances of Square, one instance of Circle, and one instance of Rectangle. Use a loop to iterate thru the array and invoke method getArea from each instance; if applicable, invoke method howToColor on each instance.

Here is a sample run:

Area is 4.0 Color all four sides

Area is 20.25 Color all four sides

Area is 25.0 Color all four sides

Area is 78.53981633974483

Area is 12.0

**Question 06 (10 points)**

Define a class named ComparableSquare that extends Square (defined above) and implements interface Comparable. Override compareTo method to compare the Squares on the basis of area. Write a test class to find the larger of two instances of ComparableSquare objects.

Here is a sample run:

Comparing the areas of two squares for sides 5 and 8

the largest Area is 64.0