CSC1016S Assignment 7: Interfaces, ADTs, and Software Testing

Assignment Instructions

This assignment concerns (i) the use interface declarations to construct abstract data types and the use of class declarations to construct concrete (implementing) sub types; (ii) constructing and performing software testing using JUnit.

Question 1

Write code that will accept as input the characteristics of an undefined number of different softdrinks. Each softdrink should have a colour (of the can), name and volume.

- Create SoftDrink objects and store them in an ArrayList as each softdrink is entered.
- Prompt the user to add a softdrink until they choose to quit.
- Once the user quits, print out a list of the softdrinks that the user has entered, sorted first by alphabetical order of name, then by colour, then by volume (ascending order).

Name your driver class `Question1.java`.

NOTE:

- You can use the built-in static `sort()` method in the `java.util.Collections` class to sort your ArrayList of SoftDrink objects.
- Do not use normal arrays.
- Click HERE and HERE for information on how to make sure your SoftDrink class will work with `Collections.sort()`.

Sample I/O

```
Enter option: (1) add soft drink (2) quit:
1
Enter name, colour and volume in ml separated by space
Fanta Orange 500
Enter option: (1) add soft drink (2) quit:
1
Enter name, colour and volume in ml separated by space
Fanta Orange 200
Enter option: (1) add soft drink (2) quit:
1
Enter name, colour and volume in ml separated by space
Coke Red 500
Enter option: (1) add soft drink (2) quit:
1
Enter name, colour and volume in ml separated by space
Coke Silver 500
Enter option: (1) add soft drink (2) quit:
2
Coke Red 500
Coke Silver 500
```

CONTINUED

```
Fanta Orange 300
Fanta Orange 500
```

Question 2

See the attached `Question2.java` source file. Its main method prompts the user to enter a list of animals and then prints out a list of corresponding animal noises once they are finished.

Write the code necessary to make `Question2.java` compile and produce the output below when run. You will need to create a MakesSound interface with a public method `makeNoise()` as well as Cat, Dog and Cow classes that implement the MakesSound interface.

Sample I/O

```
What animal do you see? (1) Cat (2) Dog (3) Cow (4) quit
1
What animal do you see? (1) Cat (2) Dog (3) Cow (4) quit
1
What animal do you see? (1) Cat (2) Dog (3) Cow (4) quit
1
What animal do you see? (1) Cat (2) Dog (3) Cow (4) quit
2
What animal do you see? (1) Cat (2) Dog (3) Cow (4) quit
1
What animal do you see? (1) Cat (2) Dog (3) Cow (4) quit
1
What animal do you see? (1) Cat (2) Dog (3) Cow (4) quit
3
What animal do you see? (1) Cat (2) Dog (3) Cow (4) quit
4
The animals want to say goodbye:
Meeow
Meeow
Meeow
Woof!
Meeow
Meeow
Moo!
```

Question 3

Write a CSC1016S class that stores student information and marks, and calculates a student's final mark. Your class must implement the attached Student interface.

The 4 methods in Student are as follows:

- `setName()` sets the name of the student.
- `getName()` returns the name of the student.
- `setMark()` sets one of 4 marks, depending on the category parameter, which is one of `{"pracs", "practests", "tests", "exam"}`.
- `getFinal()` returns the final mark using the formula from the notes to students. All calculations are done with floats.

There is no test program. Instead you will test this class using a technique called unit testing.

Unit testing is an automatic testing technique for individual modules of a large software system. Tests are defined and executed automatically to rapidly test and re-test a class during development without user intervention. This helps to scale up testing when a program gets very large. It also ensures the correctness of each individual class within the system.

JUnit is one of the most popular unit testing frameworks for Java. The advantage of a framework (rather than writing small programs to test each class) is that there is a standard API, common understanding of the testing approach among Java programmers and integrated support in many IDEs.

Learn how to use JUnit by reading the online documentation or an online tutorial.

Write JUnit tests for your CSC1016S class in a class called CSC1016STest, stored in the file CSC1016STest.java. You must include the following 4 tests.

1. A test that `getName()` returns the value set using `setName()`.
2. A test that, if all marks are 0, the final mark is 0.
3. A test that, if all marks are 100, the final mark is 100.
4. A test that the final mark is correctly calculated for all marks being between 0 and 100.

Download the JUnit packages from Resources/Software/JUnit.

JGrasp has built-in integration with JUnit.

- Go to Tools/JUnit and use Configure to first set the location of the JUnit packages you downloaded. Then there will be options to run JUnit tests from within JGrasp.
- When you add source files to a JGrasp project, JGrasp will automatically detect which ones are normal source files and which ones are JUnit test classes.

Alternatively, in order to compile your JUnit test class from the command-line, use a command such as:

```
javac -cp junit-4.12.jar:hamcrest-core-1.3.jar:. CSC1016STest.java
```

And in order to execute your JUnit tests from the command-line, use a command such as:

```
java -cp junit-4.12.jar:hamcrest-core-1.3.jar:. org.junit.runner.JUnitCore
CSC1016STest
```

CONTINUED

Marking and Submission

Submit `Question1.java`, `SoftDrink.java`, `MakeSound.java`, `Cat.java`, `Dog.java`, `Cow.java`, `CSC1016S.java` and `CSC1016STest.java` in a single .ZIP folder to the automatic marker.

The zipped folder should have the following naming convention:

yourstudentnumber.zip

END