NORTH SOUTH UNIVERSITY
Department of Electrical and Computer Engineering
B.Sc. in Computer Science and Engineering Program
Mid Term II Examination, Summer 2019 Semester

Course: CSE 215 Programming Language II, Section-14
Instructor: Mohammad Rezwanul Huq (MRH1), PhD, Assistant Professor (Part-time)
Full Marks: 40 (20 will be counted for final grading)
Time: 1 Hour and 20 Minutes

Note: There are **5** (**FIVE**) questions, answer ALL of them. Marks of each question is mentioned at the right margin.

---

1. **Define** a class named **Money** whose objects represent amounts of U.S. money. The class should have two private instance variables of type int for the *dollars* and *cents*. The Money class must have the following methods. 8

- Include appropriate *constructors*, *setters* and *getters*.
- Include *toString* method that returns a string as per the following format-
  The Money object has the value *x* dollars and *y* cents. [where x and y are the actual values of dollar and cents fields]
- Include *compareTo* method which must be invoked in associated with a caller object and takes an argument which is the callee object. The method returns 1 if the caller object has greater value than the callee object; returns 0 if both objects have the equal values; otherwise, returns -1.The following line shows the sample method definition.
  ```
  public int compareTo(Money m){ … }
  ```
- Include the methods *add* for addition of amounts of money. These methods should be invoked in associated with a caller object and takes an argument which is the callee object. The added value must be returned as another object of Money type. The following two lines show the sample method definitions.
  ```
  public Money add (Money m){ … }
  ```

Further, **define** a main method within the **Main** class and demonstrates the functionalities of Money class. You must create a couple of Money type objects and the must call the methods – *add*, *compareTo* and *toString* accordingly.

2. Assume that you are one of the programmers developing an application for NSU Student Clubs. Your job is to build a module through which the university can keep track of different activities organized by all the clubs. Please note that, a club can organize many activities whereas an activity must belong to a club. You need to do the followings to build the module. 10

Define a class **Club** that has three instance variables: *clubId*, *clubName*, *yearEstablished*. *Club* also has a static variable *count* that is used to count the total number of clubs. Include appropriate constructors. Add *toString()* method that returns a string containing club's information.

Define another class **Activity** that has three instance variables: *activityId, activityName* and *activitySponsor.* Include appropriate constructors. Also include setters and getters if required. Add *toString()* method that returns a string containing information of that activity.

Inside the **Club** class, you should also hold the activities' information which are organized by a club. Moreover, the **Club** class must have an `addActivity(Activity a)` method which is invoked when a club organizes an activity.

Write a **Main** class that includes a *main* method for testing the functionalities of these two classes: **Club** and **Activity**. Create a couple of relevant objects of both classes and invoke *addActivity* method of the *Club* class appropriately.

3. A post office has mainly two types of services: Mail service delivers mails for a postage fee and Cargo service delivers goods based on its weight known as cargo fee plus handling fee. The post office wants you to write a Java application that performs its bill calculations polymorphically considering the following specifications. 5

Service class has two instance variables: *serviceId* and *deliverAddress*. *Service* class also includes a method *bill()*.

MailService has an instance variable: *postageFee*. **CargoService** has three instance variables: *weight, cargoFee* and *handlingFee*. All classes include appropriate constructors and *toString()* method.

Draw the corresponding class diagram. Use appropriate access specifier symbol, data type and return type of methods.

4. As per the use case specified in Question 3, *bill()* and *toString()* methods return the followings: 12

Class	*bill()* Method	*toString()* Method
Service	returns 0	*serviceId deliverAddress*
MailService	returns *postageFee*	*serviceId deliverAddress* postage fee: *postageFee*
CargoService	returns the sum of *cargoFee and handlingFee*	*serviceId deliverAddress* cargo fee: *cargoFee* handling fee: *handlingFee*

Define *Service* class. Include constructor(s), toString() and a method *bill()*.

Define *MailService* class. Include constructor(s) and override bill() and toString() methods as per the specification shown in the table above.

Define *CargoService* class. Include constructor(s) and override bill() and toString() methods as per the specification shown in the table above. Handling fee is 10% of the cargo fee.

Write a *Main* class that has *main()* method. Within the *main()*, define an ArrayList of appropriate type. Instantiate two objects each from *MailService* and *CargoService* class and add those four objects into that array list. Invoke *bill()* and *toString()* methods polymorphically and print these values. At last, print the average bill of *CargoService* type objects.

5. **Match** keywords in column A with appropriate statement in column B. 5

Keywords (Column A)	*Statements (Column B)*
A. Inheritance **B.** Object **C.** Association **D.** Method Overriding **E.** Static Variable	**i.** methods having the same name with same method signatures in a superclass-subclass relationship **ii.** relationships among non-related classes **iii.** fields of a class **iv.** an entity that exists in real world **v.** variable belongs to the object **vi.** methods having the same name but different method signatures **vii.** 'is a' relationship **viii.** a general binary relationship between classes **ix.** variable belongs to the class **x.** 'has' relationship