MANIPAL UNIVERSITY JAIPUR
SCHOOL OF COMPUTING AND IT
Semester 1, Session 2017-18
B. Tech. III Semester Second Sessional Examination
Branch: CSE / IT /CCE
CS1304 Object Oriented Programming Using Java
(OPEN BOOK)

Duration: 1 hour **Max. Marks: 15**

Instructions:
- All questions are compulsory.
- Missing data if any may be assumed suitably and Specify all necessary assumptions.
- Handwritten notes are allowed but not their photocopies. Two Books or their bounded photocopies are allowed.

1 You are given an incomplete class named 'MenuItem' having three attributes named : -itemname:String, -price:String and –category:String. The class has only one constructor which receive one argument of type String which embeds the values of the three attributes separated by "$$$" as per the *format*: itemname$$$price$$$category.

(a) You have to complete the constructor of this class so that the value of all the attributes can be extracted and assigned suitably. [2]

```
class MenuItem
{
        private String itemname;
        private String price;
        private String category;
        MenuItem(String details)
        { //The constructor extract the values of various attributes
        // Q1(a) Complete this part
        }
// Assume the accessor and setter methods for all three instance variables are provided.
}// End of class MenuItem
```

(b) You are given an incomplete class named 'Menu' as follows: [1+2

```
class Menu
{
        MenuItem items[]=new MenuItem[10];
        public static int count=0;
        public void addMenu(MenuItem m)
        {   // This method adds 'm' in to items at index 'count' and updates count value.
            // If 'count' value is 10 it returns without adding
            // Q1(b-1) Complete this part
        }  //End of Method
        public void printReport()
        { // This method will print the elements of items in a length of 60 characters such that first 30
          //character will be reserved for Itemname, next 20 char for price and last 10 for category
          //Q1(b-2) Complete this part
        }//End of Method
}//End of Class Menu
```

2 Consider an interface named 'OnlineAccount' that models an online e-commerce account like Amazon prime. As follows:

```
interface OnlineAccount
{
    double baseprice=100                    //base price for the account
    double reguralSerialPrice = 20;         // price for regular serial
    double premiumSerialPrice=50;           //price for premium serial
}// End of Interface
```

(a) You are given an incomplete class named 'Account' as follows and you have to complete it as per the commented specification [1+1+2]

```
class    Account    implements    OnlineAccount, Comparable<Account>
{
        int noRegularSerial,noPremiumSerial;
        // Q2a(1) Add a parameterised constructor to initialize the variable noRegularSerial and
        //noPremiumSerial
        public double monthlyCost()
        {//This method returns monthly cost for the account by adding the base price to the product of
         //noRegularSerial and regularSerialPrice and product of noPremiumSerial and remiumSerialPrice
         // Q2a(2) Complete this part
        }
        // Q2a(3) Override compareTo method such that two accounts are compared based upon their
        //monthly cost
        public String toString()
        { //Assume suitable implementation is given }
}// End of class Account
```

(b) [1]

```
public class Demo
{
        public static void sortAccount(Account acc[])
        {//This method sorts the elements of array 'acc' in ascending order of monthly cost and displays
         // sorted result
        //Q 2(b) Complete this method
        }
} // End of class Demo
```

3 Consider an abstract class named 'Message' as follows [5]

```
abstract class Message
{
        private String message; // String type message
        Message(String message) { this.message = message; }   // Constructor Message
        public  String  getMessage()   {      return  message;        } // Accessor Method
        public  abstract String        encrypt();                 // Abstract Method
}// End of class Message
```

The 'encrypt()' abstract method is required to encrypt the string message. Complete the implementation of this method for a 'Message' sub-class named 'NumericMessage' as per commented specification given below:

```
class   NumericMessage         extends Message
{
        public          String          encrypt()
        {
         /* This Method encrypts the message and returns its encrypted form only if the message
         represents a proper numeric message. A message is a proper numeric message only if it consists
         of characters in the range 0-9. For example, "1237873" is a proper numeric message whereas
         "123rty" is not. If message does not represent a proper numeric message, then this method
         returns a 'null' value. If message represents a proper numeric message, then it returns an
         encrypted string value by swapping its adjacent elements. For example, if message is "123467"
         then it returns "214376", if message is "1234rty" then it returns 'null' as message is not in proper
         binary form. If the message is "12345" then it returns "21435" */
         //Implement This Method
         }// End of Method
}// End of class Numeric
```