CSE213, Object Oriented Programming
Independent University, Bangladesh
Midterm(AUTUMN)-2020

1. Write COMPLETE java program (Test.java) for the following incomplete code: [Only ONE JAVA FILE]

```
class Complex {
int real, img;
//add constructor and other necessary methods for input, output
} //represents complex number in the form of “real +/- img i” e.g: 2+3i
class ComplexList {
ArrayList<Complex> cList;
//write populate & show methods

}
//rewrite TEST class with full functioning code
public class Test {
public static void main(String[] args){
ComplexList list;
//....
do-while loop: as long as user wants [choice: 1 to add, 2 to exit]
{
list.add(new Complex().setComplex());
}
int n = s.nextInt(); //no of additional Complex numbers
Complex[] cArr = new Complex[n];
List.augment(cArr);
//augment method will instantiate cArr with n Complex instances,
//set their values, and finally merge cArr with cList of
//ComplexList instance list
int lower, upper;
//get values of lower & upper from user
List.display(lower,higher);
//display all the Complex numbers from cList of list
//whose real<=lower & img>=upper, using showCopmlex()
//method of Complex class;
}
}
```

P.T.O

1

2. Write COMPLETE java program for the following incomplete code (Test.java): [Only ONE JAVA FILE]

```
class MyArray {
int[] intData;
// add necessary methods
}
class Matrix {
MyArray[ ][ ] oneDObjects;
// add necessary methods ( see main() )
}
public class Test {
public static void main(){
Matrix m = new Matrix();
Loop: as long as user wants to continue
{
m.populateAndAugment();
// get values of # of rows & cols from user and instantiate
// oneDObjects. Now ask size of each MyArray instances of
// oneDObjects and populate them with random integers.
// If oneDObjects is already populated, then ask for additional
row
// & col size and do the above for additional MyArray instances
// to fill the new cells (additional elements of oneDObjects) of
the matrix
m.display();
// display the Matrix of MyArray instances
}
}//end main()
}//end class
```

P.T.O

2

3. Create a NetBeans java CONSOLE project with two packages: myarrays and mainpkg

- Classes of your projects are: myarrays.OneDArray, myarrays.Matrix and mainpkg.MainClass
- Class OneDArray has following private fields: int[] values, float average
  - Methods: a) void getArray(); b) void showArray();
  - If necessary, add other methods to ensure that your main method works
- Class Matrix has following private fields: OneDArray[][] arrays
  - According to given RUN, you need to add appropriate methods in Matrix class

MainClass has the following main method:
public static void main(String[] args){

Matrix m1, m2, m3;
r = no of rows for Matrix class object. r is a user input
c = no of columns for Matrix class object. c is a user input
m1 = new Matrix(r, c); // m1 will have r rows & c cols
//stores OneDArray instances in arrays[i][j] inside m1 Matrix instance
//ask user for length and values for each OneDArray
//average value of each MyOneDArray is also calculated
Sout("First Matrix:"); m1.showMatrix(); //see RUN
m2 = new Matrix(r, c, 2, 10); // m2 will have r rows & c cols
//3
rd parameter is the length of first OneDArray in m2,
//which gets incremented by 1 for subsequent OneDArray instances in m2
//4
th parameter is the upper limit of random values to populate m2
//average value of each OneDArray is also calculated
Sout("Second Matrix:"); m2.showMatrix();//see RUN
m3 = m1.merge(m2);
Sout("Merged Matrix:"); m3.showMatrix();//see RUN
}
RUN:
How many rows? 2
How many columns? 2
How many numbers: 2
Enter values: 1 3
How many numbers: 3
Enter values: 4 8 6
How many numbers: 2
Enter values: 7 2
How many numbers: 4
Enter values: 3 7 9 1

First Matrix:
{1,3} Avg: 2 {4,8,6} Avg: 6
{7,2} Avg: 4.5 {3,7,9,1} Avg: 5
Second Matrix:
{5,1} Avg: 3 {3,1,7} Avg: 3.67
{9,1,5,4} Avg: 4.75 {2,7,1,8,5} Avg: 4.6
Merged Matrix:
{1,3,5,1} Avg: 2.5 {4,8,6,3,1,7} Avg: 4.83
{7,2,9,1,5,4} Avg: 4.67 {3,7,9,1,2,7,1,8,5} Avg: 4.78

Now,

● Implement the above project without changing main(), fields of the classes and given RUN