Object Oriented Programming

CSE 1115/CSI 211 (D)

Section: D

Md. Fahim Bin Amin

ID: 011201158

Ans. to the Q:N: 01

```
import java.io.*;
import java.util.ArrayList;

public class Main {
 public static void main (String [ ] args) {
 ArrayList <string> l = new ArrayList < > ();
 try {
        FileReader fr = new FileReader ("in.txt");
        File Writer fw = new FileWriter ("out.txt");
        BufferedReader br = new BufferedReader (fr);
        BufferedWriter bw = new BufferedWriter (fw);
        string line;
        while ((line = br.readLine()) != null) {
              l.add (line);

        }
    for (int i = l.size() - 5; i < l.size(); i++) {
        bw.write (l.get (i) + "\n");
    }
```

```
    bw.close();
  } catch(IOException e) {
      e.printStackTrace();
  }
}
}
```

Ans. to the Q: N: 2 (A)

```
public class ExceptionTest {
 public static void main (String[ ] args) {
try {
 Scanner sc = new Scanner (System. In);
 int data [ ] = new int [10];
 for (int i = 0; i <10; i++) {
  int a = sc. nextInt ();
  int b = sc.nextInt ();
  int c = sc.nextInt ();
  data [a] = b/c;
 }
} catch (InputMismatchException e) {
  System.out.println ("Input mismatch");
} catch (ArrayIndexOutOfBoundsException e) {
  System. out. println ("Out of Index");
}
```

```
catch (ArithmeticException e) {
 System.out.println("cannot divide by 0");
}
finally {
   System.out.println("Done!");
}.

}
```

An. to the Q:No 2(b)

```
class Account{
    String name;
    double balance;
    Account (String name, double balance){
        this.name = name;
        this.balance = balance;
    }
    public void withdraw(double amoun) throws
    InsufficientBalance{
        if (amount < balance)
            balance = balance - amount;
        else
            throw new InsufficientBalance(balance,
                                          amount);
    }
}
```

```
class InsufficientBalance extends Exception {
public InsufficientBalance(double curBalance,
                           double withdraw) {
super("Insufficient Balance. Current balance"
  + curBalane + " is lower than the withdraw
  amount " + withdraw);

}
}
public class Main {
public static void main(String [] args) {
try { ~~Account ac =~~
  Account ac = new Account("Julin", 10000);
  ac.withdraw(11000);
  System.out.println(ac.balance);
} catch (InsufficientBalance e) {
  System.out.println(e.getMessage());
}
}
}
```

Answer to the Q.N: 03 (Part -A)

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;

public class Main {
public static void main (String[] args){
ArrayList <Movie> list = new ArrayList <>();
list.add (new Movie ("Walking Dead", 8.3));
list.add (new Movie ("Avengers", 7.3));
list.add (new Movie ("Pothen Pachali", 9.5));
list.add (new Movie ("Your Name", 6.1));
list.add (new Movie ("Inception", 9.8));
Collections.sort(list, new CompareRating());
for(int j=0, j < list.size(); j++) {
    System.out.println(list.get(j).name + " "
            + list.get(j).imdbRating);
    }
  }
}
```

```
class Movie {
    String name;
    double imdbRating;
  public Movie(String name, double imdbRating) {
    this.name = name;
    this.imdbRating = imdbRating;
  }
}

class CompareRating implements Comparator<Movie>{
  public int compare(Movie o1, Movie o2){
  if (o1.imdbRating < o2.imdbRating) return 1;
  else if (o1.imdbRating > o2.imdbRating) return -1;
  return 0;
  }
}
```

Ans. to the Q.N: 4 (A)

Code - I

We need to modify the method, secondMethod ( )

After the modification, the correct code would be,

```
void secondMethod ( ) {
    System. out.println("b = " + b);
    System.out. println ("c = " + c);
    this. a = 5;
}
```

Reason:

@ at the end of the code, this.a cannot be referenced from a static content.

Code - II

After the modification, the code would be:

```
class A {
        int x = 0
        ---
        ---
}
```

reason for providing class A instead of final class A: We can not extend the final class. If we need a child class of class A, then class A must not be final. And for int x, ~~th~~ the code was expecting an identifier.

```
class C {
    void anotherMethod() {
        - - - -
    }
}
```

reason for providing void anotherMethod() instead of final void anotherMethod() : We can

not override ~~an~~ a final method. If we also want to override the ~~anoth~~ anotherMethod( ) through a child class of C, named ~~D~~ class D, the method must not be final.

Ans. to the Q: N: 4(B)

1st Error:

account.updateBalance("0")

Here, we need to pass a double value instead of a string.

2nd Error:

account.updateBalance(0)

Here, we need to pass a double value instead of an integer. Like,

updateBalance(0.0)

3rd Error:

updateBalance() can not be declared here because it is not an abstract method.

Ans. to the Q: N: 5

In the line number 9 and 12, there is a common error and that is "set" method. We need to replace it with setText( ).

In the line no 19, there is another error and that is setView( ). It would be replaced by setVisible( )

So, the necessary codes:

```
frame.add(upButton);
frame.add(middleLabel);
frame.add(downTextField);
```

Ans. to the Q:N: 06

```
interface Human{
    abstract void study ( );
    abstract void work ( );
  }

class Doctor implements Human{
    public void study ( ){
    System.out.println ("Studying Human Anatomy");
  }
  public void work ( ){
   System. out. println ("Dissecting Human Body");
  }
}

class CSEngineer implements Human{
  public void study ( ){
   System. out. println ("Solving ODE/PDE");
  }
```

```
public void work () {
 System. out. println (" Coding in Java ");
 }
}
class CivilEngineer implements Human {
public void study () {
 System.out.println ("Solving ODE/PDE ");
 }
public void work () {
 System.out.println("Designing Structures");
 }
}
```