Concordia University
Department of Computer Science & Software Engineering

19.25
20

Comp 249, Winter 2017
Instructor: Aiman Hanna
Time: 60 minutes

Mid-term Exam

Name [redacted]

ID. # [redacted]

V. good

- **Keep your answer very organized & clean.**
- **Exam is closed-book; ENCS calculators are allowed. Exam booklet has 9 pages.**
- **Exam will be marked out of 20.**

4

Question # 1 (4 marks)
Indicate whether each of the following statements is true or false:

A) In Java, it is possible to throw an exception, catch it, then re-throw that same exception if desired.

✓ [✓ ΔTrue] FalseΔ

B) In order for polymorphism to work, it is sometimes crucial to create an abstract base (parent) class that has at least one private method.

✓ ΔTrue [✓ FalseΔ]

C) If a base class has a method such as: public int fun1(), an inherited class cannot reduce accessibility to this method by declaring it as protected (i.e. protected int fun1()).

✓ [✓ ΔTrue] FalseΔ

D) Assume class ***B*** is inherited from class ***A***. The following statement, where ***a1*** and ***b1*** are references to objects from ***A*** and ***B*** respectively, would result in *compilation* error: a1 = (A)b1;

A ← B

✓ ΔTrue [✓ FalseΔ]

E) A constructor of an inherited class has the right to call the **this()** constructor and the **super()** constructor but **super()** must be called first.

✓ ΔTrue [✓ FalseΔ]

F) Even if a class ***A*** is created as an abstract class, the following code will still be okay: ***A a1;***

✗ ΔTrue [✓ FalseΔ]

1

Question # 2 (3 marks)
Select the correct answer (Circle **only one** answer):

A) Given the following code (where the function ***myFun()*** and the ***main()*** method are inside one class in the same file where classes ***X***, ***Y*** and ***Z*** are):

```
class X{
      protected int i;
      public X() {
            i = 1000;
      }
      private void fun1() {
            System.out.println("Point 1");
      }
}

class Y extends X {
      void fun1() {
            System.out.println("Point 2");
            i = 2000;
            System.out.println("i is: " + i);
      }
}

class Z extends Y {
      protected void fun1() {
            System.out.println("Point 3");
            i = 3000;
            System.out.println("i is: " + i);
      }
}
```

```
public static void myFun(Object
obj1)
      {
            Z z1 = (Z) obj1;
            z1.fun1();

            Y y1 = z1;
            y1.fun1();

      }


public static void main(String[]
args)
{

      Z zobj = new Z();
      myFun(zobj);

}
```

✓ [✓ i)] The output of this code is:
```
Point 3
i is: 3000
Point 3
i is: 3000
```

ii) The output of this code is:
```
Point 3
i is: 3000
Point 2
i is: 2000
```

iii) The output of this code is:
```
Point 3
i is: 3000
Point 2
i is: 2000
Point 1
i is: 1000
```

iv) Program will not compile due to access rights problems with *fun1()* method.

v) Program will compile but will have a run-time error due to an incorrect casting.

B) Consider the following code:

```
class B extends A {

}

class C extends B {

}

class D extends B {

}
public class MyClass {

        public static void fun1(A a1) {
              C c1 = new C();
              if (a1.equals(c1))
                    System.out.print("They are equal");
              else
                    System.out.print("They are NOT equal");
        }
        public static void main(String[] args) {
              A a1 = new C();
              fun1(a1);

        }

}
```

What is the result of compiling/running the code?

X [✓ I.] The program will display : `They are equal`
II. The program will display : `They are NOT equal`
III. The code will not compile: classes have no constructors.
IV. The code will not compile: classes did not define *equals()* method.
V. The code will compile correctly, but the program will crash at run-time for Null Pointer Exception.

C) In Java I/O, the end of binary file can be detected:
I. When the method hasNextLine() returns false.
II. When an exception is thrown.
X [✓ III.] When a method returns a special value that is either null or -1 depending on the contents.
IV. When the method hasNextObject() returns an empty object.
V. Three of the above answers are correct.

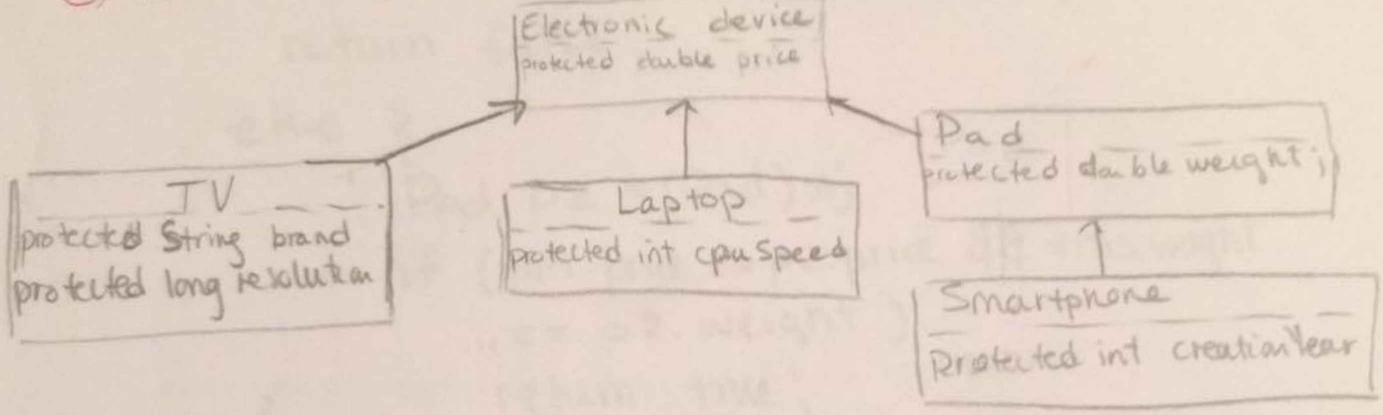
6

Question # 3 (6 marks)

The following classes are defined: **ElectronicDevice, TV, Laptop, Pad,** and **SamrtPhone** (which is considered as a type of a Pad). **ElectronicDevice** has one attribute called **price** (double type). A **TV** has two attributes called **brand** (String type), and **resolution** (long type). A **Laptop** has one attribute called **cpuSpeed** (int type), a **Pad** has one attribute called **weight** (double type), and a **SamrtPhone** has one attribute called **creationYear** (int type). Assume that all attributes in base class(es) are created as *protected.*

1

A) Analyze the above classes carefully then propose **through UML (or a small sketch/drawing)** (do **NOT** describe in writing) a suitable structure/design of these classes. (i.e. how they relate to each other).



B) You are required to write the code of the following two methods (do **NOT** write any other parts of the classes):

2

i) Assuming that the **ElectronicDevice** class has a parameterized constructor that accepts one value to initialize the *price* of the object. Write a parameterized constructor of the **TV** class. This constructor however **must** take advantage, *if possible,* of the already-written **ElectronicDevice** constructor.

✓

```
public TV(double price, String price, long resolution){
      super(price);
      this.price = price;
      this.resolution = resolution;

}
```

Question 3 Continues Next Page

Question 3 – Continues ….

3 ii) Write the ***equals()*** method of the **Pad** class, which is used to determine whether or not two **Pad** objects are equal (having the same values of all attributes). Your method must be written properly so that it overrides the default *equals()* method provided by the language.

```
public boolean equals (Object x){
  if (x==null || x.getClass()!=this.getClass())
    return false;
✓ else {
      Pad p2 =(Pad)x;
      if (this.price == p2.price && this.weight
           == p2.weight )
          return true;
   }
}
```

Question # 4 (4 marks)
Given the following code where classes *A*, *B* and *Q4* are in the same package. Will this code cause compilation or run-time error(s)? Notice that if the program compiles and runs, then there is no error; regardless of the logic of the program. If *errors exist*, indicate the exact error(s), correct them and provide the output. If *no* errors exist, provide the exact output of the program.

```
class A
{
        int x1;
        protected int x2;
        public A()
        {
                System.out.println("Executing default constructor of A");
                x1 = 0;
                x2 = 0;
        }

        public A(int i1, int i2)
        {
                System.out.println("Executing parameterized constructor of A");
                x1 = i1;
                x2 = i2;
        }
}

class B extends A
{
        public B()
        {
                super(50, 60);
                System.out.println("Executing default constructor of B");
        }

public B(int i1)
        {
                System.out.println("Executing single-parameterized constructor of B");
                x1 = i1;
        }


        public B(int i1, int i2)
        {
                System.out.println("Executing double-parameterized constructor of B");
                x1 = i1;
                x2 = i2;
        }
}
public class Q4{
        public static void main(String[] args)
    {
          A a1 = new A();
          B b1 = new B(), b2 = new B(45, 80);
          A a2 = new B(90);
    }
}
```

Answer of Question 4 Goes to Next Page

Answer of Question 4

4

Executing default constructor of A.
Executing parameterized constucter of A.
Executing defaut constructor of B.
Executing default constructor of A.
Executing double-parameterized constructor of B
Executing default constructer of A.
Executing single-parameterized constructer of B.

✓

Question # 5 (5 marks)

4.3

The collection of antique sunglasses is kept by an antique dealer at all times in a file called ***Sunglasses.txt***. However, depending on the inventory, the file may include zero or many records. Each of the records indicates the type of the sunglasses (round, oval, square, etc.), its year of creation and its price. These values are separated by tabs (i.e. there is a tab character between each of them). We are interested in tracing all *round* sunglasses and obtain the total value of these glasses in the inventory. Those sunglasses must be recorded in a file called ***RoundGlasses.txt***. To be precise, the file will include all the records of round sunglasses from ***Sunglasses.txt***, as well as a final line that includes the total of these glasses (sum of all their values).

You are required to give only particular parts of the code. In specific, i) you need to show how the two files (***Sunglasses.txt*** and ***RoundGlasses.txt***) are opened for input and output respectively, using the **Scanner** and **PrintWriter** classes. ii) You are also required to write the code of a method called ***RecordRoundGlasses()***, which accepts two parameters, a **Scanner** object representing an input stream, and a **PrintWriter** object representing an output stream, The method must then find all records in the input file that have **round** glasses. These entries must then be recorded in the output file. A final line indicating the total value of these glasses must be recorded at the end of the output file (***RoundGlasses.txt***). The method must guarantee upon its termination that the records are stored in the output file. As a requirement, when you read the records on each line, you **must** read each of the values (type, year and price) separately. You are **NOT** allowed to read the records as a whole line then break these values.

For instance, given the following input file shown below, a call to the method will result in the creation of the output file, ***RoundGlasses.txt***, as shown below. **Notice that this is just an example your solution must NOT be based on it, or refer to it.**

Sunglasses.txt			***RoundGlasses.txt***		
Round	1936	844.55	Round	1936	844.55
Oval	1944	433.99	Round	1922	392.97
Round	1922	392.97	Round	1960	729.75
Round	1960	729.75	Round	1973	399.16
Square	1988	129.90	Round	1936	700.09
Rectangle	1955	230.00			
Irregular	1910	1329.55	Total Value of Round Glasses: 3066.52$.		
Round	1973	399.16			
Round	1936	700.09			

Here are the needed imports

```
import java.util.Scanner;
import java.io.PrintWriter;
import java.io.FileOutputStream;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
```

Answer of Question 5 Goes to Next Page

Answer of Question 5 Answer:

i) **Show how the input and output streams will be created, then call the *RecordRoundGlasses()*, with the two streams as parameters.**

```
Scanner sc=null;
PrintWriter pw=null;

try{
     RecordRoundGlasses (sc,pw); }
1.75
catch (FileNotFoundException e){
   System.out.println(e.getMessage();}.
catch (IOException e){
  System.out.println (e.getMessage();}.
```

ii) Give the code of the *RecordRoundGlasses()* method

```
public void RecordRoundGlasses (Scanner sc2 ,PrintWriter pw2){
    Scanner sc2 =new Scanner(new FileInputStream("Sunglasses.txt"
    PrintWriter pw2= new PrintWriter(new FileoutputStream("RoundGlasses.txt
   double totprice =0;
   String type = null;
   int  year =0;
  double price =0;
 while (sc2.hasNextLine()){
 type=sc2.nextLine().useDelimiter("//s");
      if (type.equals("Round")){
2.[illegible]5    year = sc2.nextInt();
          price= Sc2.next Double ();
          totprice = totprice +price ;
          pw2. println (type+" "+ year+" "+ price );
          if (!sc2.hasNextLine())
             pw2.println ("Total value of round glasses: "+totpric
      else sc2nextLine(); }
    else sc2.nextLine(); } }
```