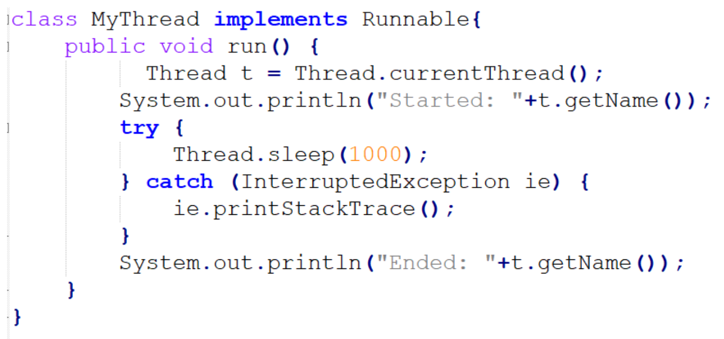
United International University
CSI 211: Object Oriented Programming, Fall 2017
Final Exam
Total Marks: 40, Time: 2 hours

Answer any 5 out of the following 6 questions $(5 \times 8 = 40)$**.**

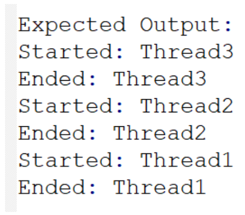
1. (a) For the class provided in Fig. 1, write a Test class to get the Expected Output. You are not allowed to remove any line of code from MyThread class. [4]

   (b) What will be the output of the java program in Fig. 2? [1]

   (c) The program in Fig. 3 has some errors. Rewrite the corrected code and underline where you have made corrections. You are not allowed to delete any line, but you may add necessary codes. [3]

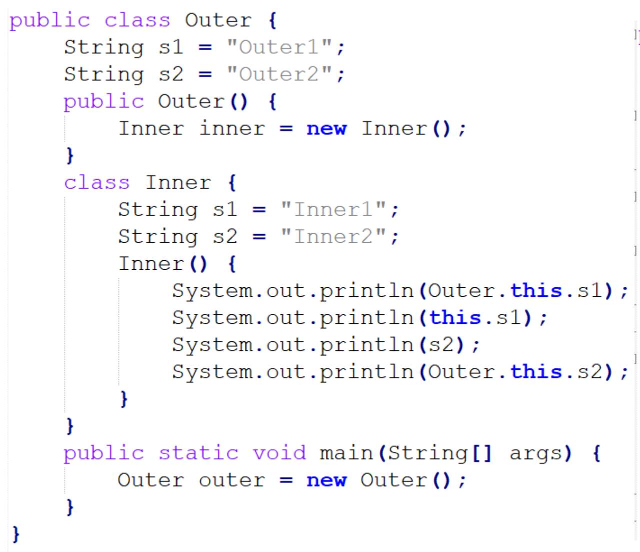


```java
class MyThread implements Runnable{
    public void run() {
          Thread t = Thread.currentThread();
        System.out.println("Started: "+t.getName());
        try {
            Thread.sleep(1000);
        } catch (InterruptedException ie) {
            ie.printStackTrace();
        }
        System.out.println("Ended: "+t.getName());
    }
}
```



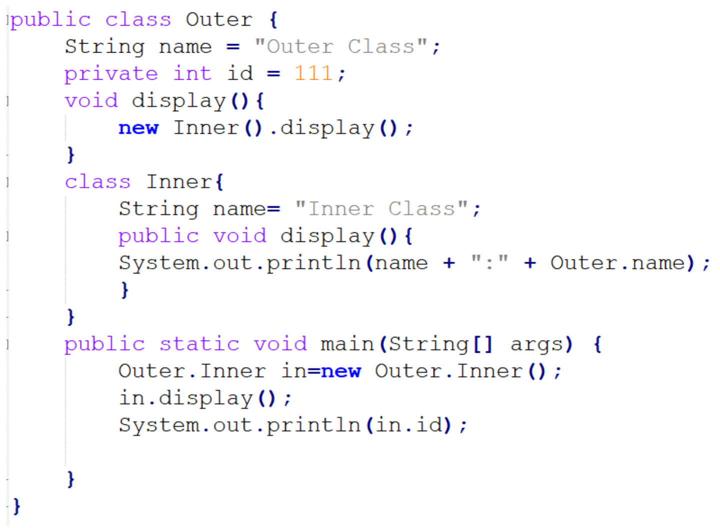
```
Expected Output:
Started: Thread3
Ended: Thread3
Started: Thread2
Ended: Thread2
Started: Thread1
Ended: Thread1
```

Figure 1: Q. 1(a)



```java
public class Outer {
    String s1 = "Outer1";
    String s2 = "Outer2";
    public Outer() {
        Inner inner = new Inner();
    }
    class Inner {
        String s1 = "Inner1";
        String s2 = "Inner2";
        Inner() {
            System.out.println(Outer.this.s1);
            System.out.println(this.s1);
            System.out.println(s2);
            System.out.println(Outer.this.s2);
        }
    }
    public static void main(String[] args) {
        Outer outer = new Outer();
    }
}
```

Figure 2: Q. 1(b)



```java
public class Outer {
    String name = "Outer Class";
    private int id = 111;
    void display(){
        new Inner().display();
    }
    class Inner{
        String name= "Inner Class";
        public void display(){
        System.out.println(name + ":" + Outer.name);
        }
    }
    public static void main(String[] args) {
        Outer.Inner in=new Outer.Inner();
        in.display();
        System.out.println(in.id);

    }
}
```

Figure 3: Q. 1(c)

2. (a) Consider a class, *MyClass* which has only three member variables: String *s*, int *i*, double *d*. Now write the following two methods: [5]

   i. *MyClass readFromFile(String fileName)*: This method reads from the file named *fileName*, creates an object and returns it. The file contains three lines: a line for string, a line for int and a line for double. After reading these lines, this method creates a *MyClass* object using these information and returns it.

   ii. *void writeInFile(String fileName, MyClass obj)*: This method writes three lines into the file named *fileName*. The three lines are for the values of three member variables of the object *obj*.

   (b) Write a generic method *minimum* that takes three input parameters. The parameters are objects of same generic type. The generic type has an upper bound: a *Comparable* interface of same generic type. The method returns the minimum object. N.B. Provide ONLY the generic method. It is not necessary to write the whole program. You must handle potential exceptions. [3]

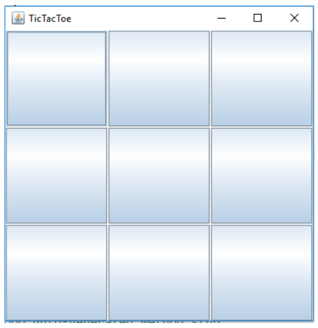
3. (a) For the following main method, add the classes Book(title, author, price) and Person(name, age) such that running the program provides the expected output. The books are sorted according to the *non-increasing* order of author's age. If multiple books have authors with same age, tie-breaking is done by the book's title. [4]

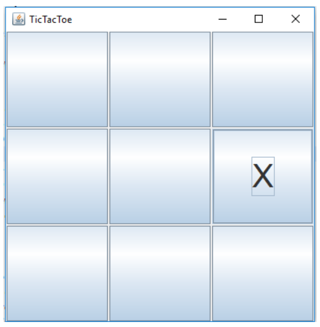
```
public static void main(String[] args) {
    ArrayList<Book> books = new ArrayList<Book>();
    books.add(new Book("The Shining", new Person("Stephen King", 50), 50));
    books.add(new Book("1984", new Person("George Orwell", 70), 40));
    books.add(new Book("The Great Gatsby", new Person("F. Scott Fitzgerald", 80), 70));
    books.add(new Book("Manhattan Beach", new Person("Jennifer Egan", 70), 100));

    Collections.sort(books);

    for (Book b : books) {
        System.out.println(b);
    }
}
```

Expected Output:

```
The Great Gatsby by F. Scott Fitzgerald : 70
1984 by George Orwell : 40
Manhattan Beach by Jennifer Egan : 100
The Shining by Stephen King : 50
```

(b) Create a simple 3x3 TicTacToe GUI as shown below. Initially the game board will consist of 9 blank cells. Whenever a user clicks on a cell, a *X* mark should be placed in the cell. [4]



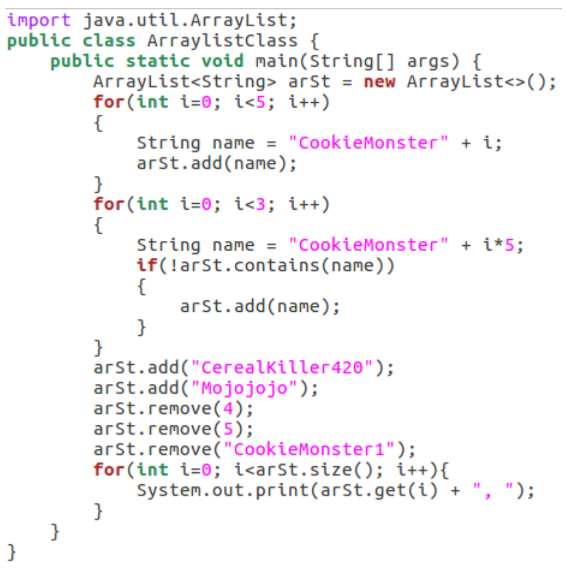


4. (a) Consider the program in Fig. 4:

   i. Find the output of the program. [2]

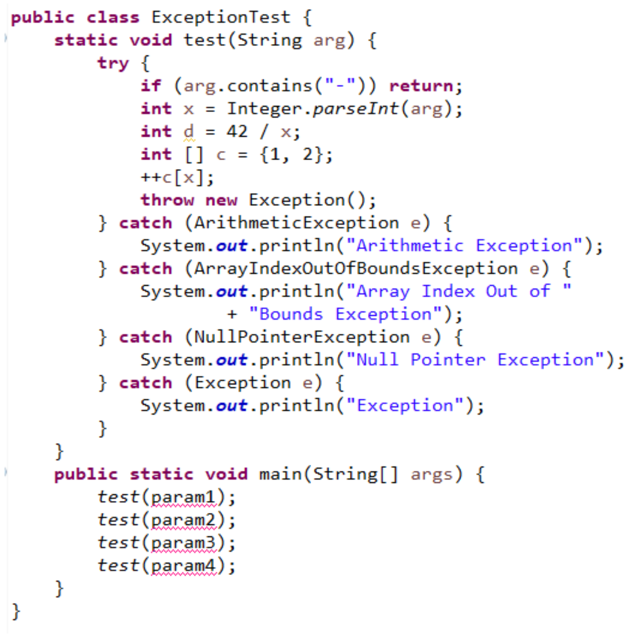
   ii. Now, you want to edit one of the elements of the given ArrayList *arSt*. Assume that you only know the ArrayList *arSt*. You want to change the element "CerealKiller420" to "CerealKiller400". Write only the code snippet to do this change. Assume that the element "CerealKiller420" is definitely present in *arSt*. [2]

(b) Provide valid values for the parameters of *test* in *main* method such that the program in Fig. 5 catches: (i) ArithmeticException for *param1*, (ii) ArrayIndexOutofBoundsException for *param2*, (iii) NullPointerException for *param3*, and (iv) Exception for *param4*. [4]



```java
import java.util.ArrayList;
public class ArraylistClass {
    public static void main(String[] args) {
        ArrayList<String> arSt = new ArrayList<>();
        for(int i=0; i<5; i++)
        {
            String name = "CookieMonster" + i;
            arSt.add(name);
        }
        for(int i=0; i<3; i++)
        {
            String name = "CookieMonster" + i*5;
            if(!arSt.contains(name))
            {
                arSt.add(name);
            }
        }
        arSt.add("CerealKiller420");
        arSt.add("Mojojojo");
        arSt.remove(4);
        arSt.remove(5);
        arSt.remove("CookieMonster1");
        for(int i=0; i<arSt.size(); i++){
            System.out.print(arSt.get(i) + ", ");
        }
    }
}
```

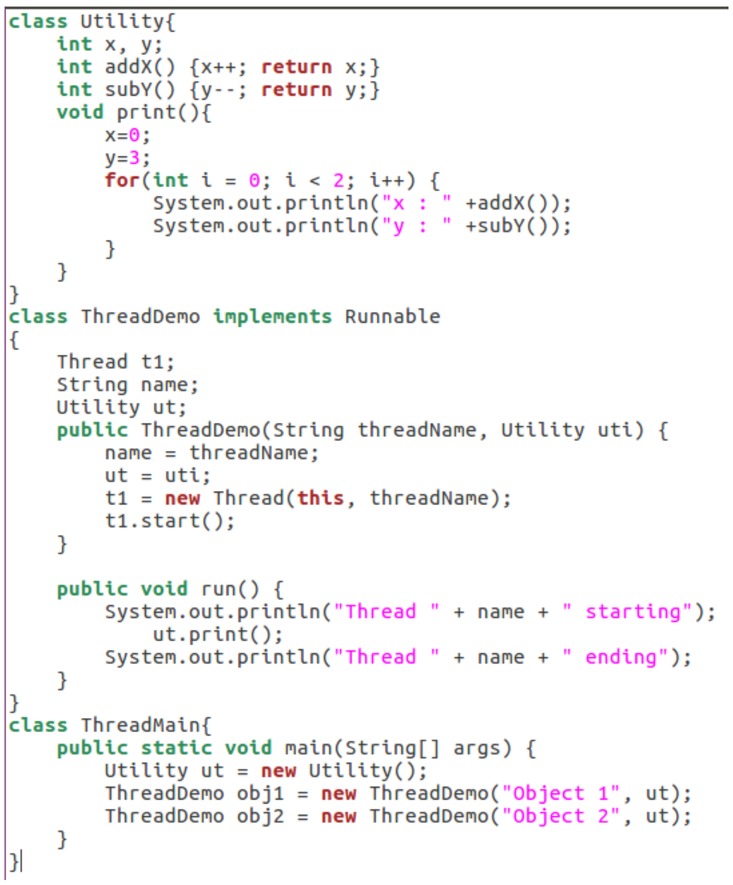
Figure 4: Q. 4(a)



```java
public class ExceptionTest {
    static void test(String arg) {
        try {
            if (arg.contains("-")) return;
            int x = Integer.parseInt(arg);
            int d = 42 / x;
            int [] c = {1, 2};
            ++c[x];
            throw new Exception();
        } catch (ArithmeticException e) {
            System.out.println("Arithmetic Exception");
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Array Index Out of "
                + "Bounds Exception");
        } catch (NullPointerException e) {
            System.out.println("Null Pointer Exception");
        } catch (Exception e) {
            System.out.println("Exception");
        }
    }
    public static void main(String[] args) {
        test(param1);
        test(param2);
        test(param3);
        test(param4);
    }
}
```

Figure 5: Q. 4(b)

5. (a) Consider the following three classes: *Utility*, *ThreadDemo* and *ThreadMain*. What will be the output of the code? [3+1]

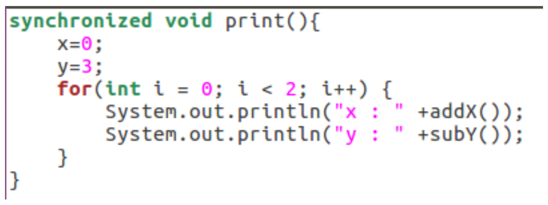


```java
class Utility{
    int x, y;
    int addX() {x++; return x;}
    int subY() {y--; return y;}
    void print(){
        x=0;
        y=3;
        for(int i = 0; i < 2; i++) {
            System.out.println("x : " +addX());
            System.out.println("y : " +subY());
        }
    }
}
class ThreadDemo implements Runnable
{
    Thread t1;
    String name;
    Utility ut;
    public ThreadDemo(String threadName, Utility uti) {
        name = threadName;
        ut = uti;
        t1 = new Thread(this, threadName);
        t1.start();
    }

    public void run() {
        System.out.println("Thread " + name + " starting");
            ut.print();
        System.out.println("Thread " + name + " ending");
    }
}
class ThreadMain{
    public static void main(String[] args) {
        Utility ut = new Utility();
        ThreadDemo obj1 = new ThreadDemo("Object 1", ut);
        ThreadDemo obj2 = new ThreadDemo("Object 2", ut);
    }
}
```

If we change the *print* method in the class *Utility* to the method in Fig. 6, will the output change? If so, what would be the new output?.

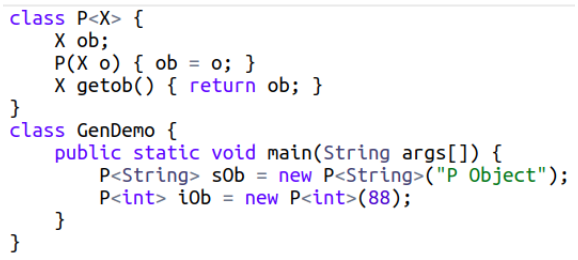
(b) What are the different stages in the life cycle of a thread? What is the method that starts the execution of a thread? [2]

(c) Consider the two classes *P* and *GenDemo* in Fig. 7. There is a problem in the *main* method in the class *GenDemo*. What is the problem, why is it a problem and what could you do to fix it? [2]



```
synchronized void print(){
    x=0;
    y=3;
    for(int i = 0; i < 2; i++) {
        System.out.println("x : " +addX());
        System.out.println("y : " +subY());
    }
}
```

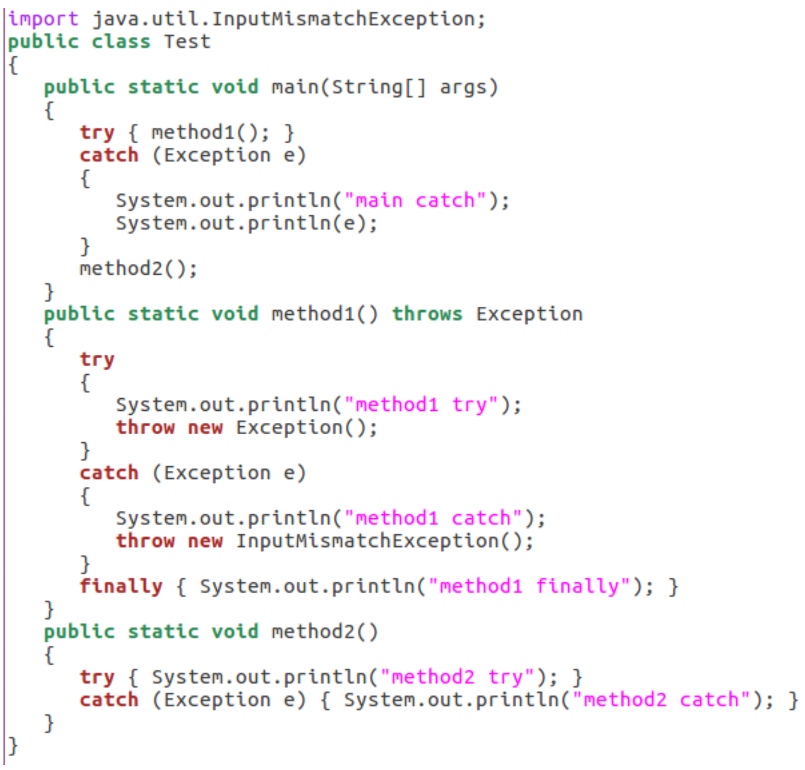
Figure 6: Q. 5(a)



```
class P<X> {
    X ob;
    P(X o) { ob = o; }
    X getob() { return ob; }
}
class GenDemo {
    public static void main(String args[]) {
        P<String> sOb = new P<String>("P Object");
        P<int> iOb = new P<int>(88);
    }
}
```

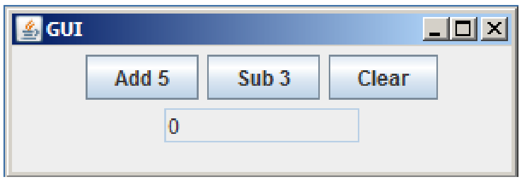
Figure 7: Q. 5(c)

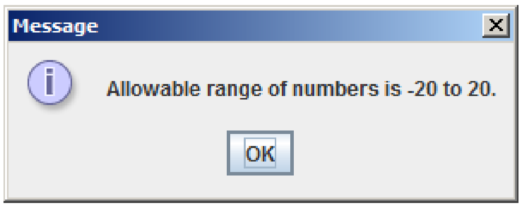
6. (a) Find out the output of the following program. [4]



```
import java.util.InputMismatchException;
public class Test
{
   public static void main(String[] args)
   {
      try { method1(); }
      catch (Exception e)
      {
         System.out.println("main catch");
         System.out.println(e);
      }
      method2();
   }
   public static void method1() throws Exception
   {
      try
      {
         System.out.println("method1 try");
         throw new Exception();
      }
      catch (Exception e)
      {
         System.out.println("method1 catch");
         throw new InputMismatchException();
      }
      finally { System.out.println("method1 finally"); }
   }
   public static void method2()
   {
      try { System.out.println("method2 try"); }
      catch (Exception e) { System.out.println("method2 catch"); }
   }
}
```

(b) Create a simple GUI application as shown below. There are 3 Buttons and 1 TextField. Initially the TextField shows 0. Clicking the "Add 5" button will add 5 to TextField value. Clicking the "Sub 3" button will subtract 3 from the TextField value. Clicking the "Clear" button will set the TextField value to 0. The range of allowed values in the TextField is -20 to 20. If at any point of pressing the buttons make the result out of range (i.e. less than -20 or greater than 20), a pop-up message will display as shown below. [4]





Allowable range of numbers is -20 to 20.