Quiz-3

Q1) Write a JUnit test class that tests the following Class. [2 marks]

```
public class Reverse{
   public static int reverseNum(int number) {
     int reverse = 0;
     while(number != 0)
     {
        int remainder = number % 10;
        reverse = reverse * 10 + remainder;
        number = number/10;
     }
     return reverse;
   }
}
```

Ans:

```
public class ReverseTest{
@Test
   public void testReverse() {
        assertEquals(Reverse.reverseNum(201),102);
   }
}
```

+0.5 for @Test
+0.5 for creating the test case function
+0.5 for calling reverseNum function appropriately
+0.5 for using assertEquals/assertTest/assertSame

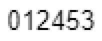
Q2) By Extending Thread Class, make a new class named ‘ThreadExtended’, which displays the thread name 10 times in its run function, delaying by a second each time. Create two threads of this class in the main function and start them. [3 marks]

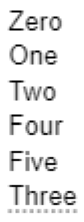
Ans:

```
public class ThreadExtended extends Thread {
   public void run() {
      for(int i=0; i<10; i++) {
         System.out.println(Thread.currentThread() + " is running now\n");
         try {
            Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
    public static void main(String[] args) {
        ThreadExtended th1 = new ThreadExtended();  // line 1
        th1.start();                                // line 2
        ThreadExtended th2 = new ThreadExtended();  // line 3
        th2.start();                                // line 4
    }
}
```

+1 for printing Thread.currentThread() (-0.5 if looping not done)
+1 for Thread.sleep (-0.5 if exception not handled)
+0.5 for creating two threads using ThreadExtended
+0.5 for th1.start() and th2.start()
(deduct 0.5 if join is called)

Q3) We have two files Digits.txt and DigitToWord.txt. Digits.txt contains a string of numerical digits. We need to read the digits one by one and write their spellings in the DigitToWord.txt file.
For example,

Digits.txt 
012453

DigitToWord.txt 
Zero
One
Two
Four
Five
Three

Write a code for accomplishing this.
[4 mark]

Ans:

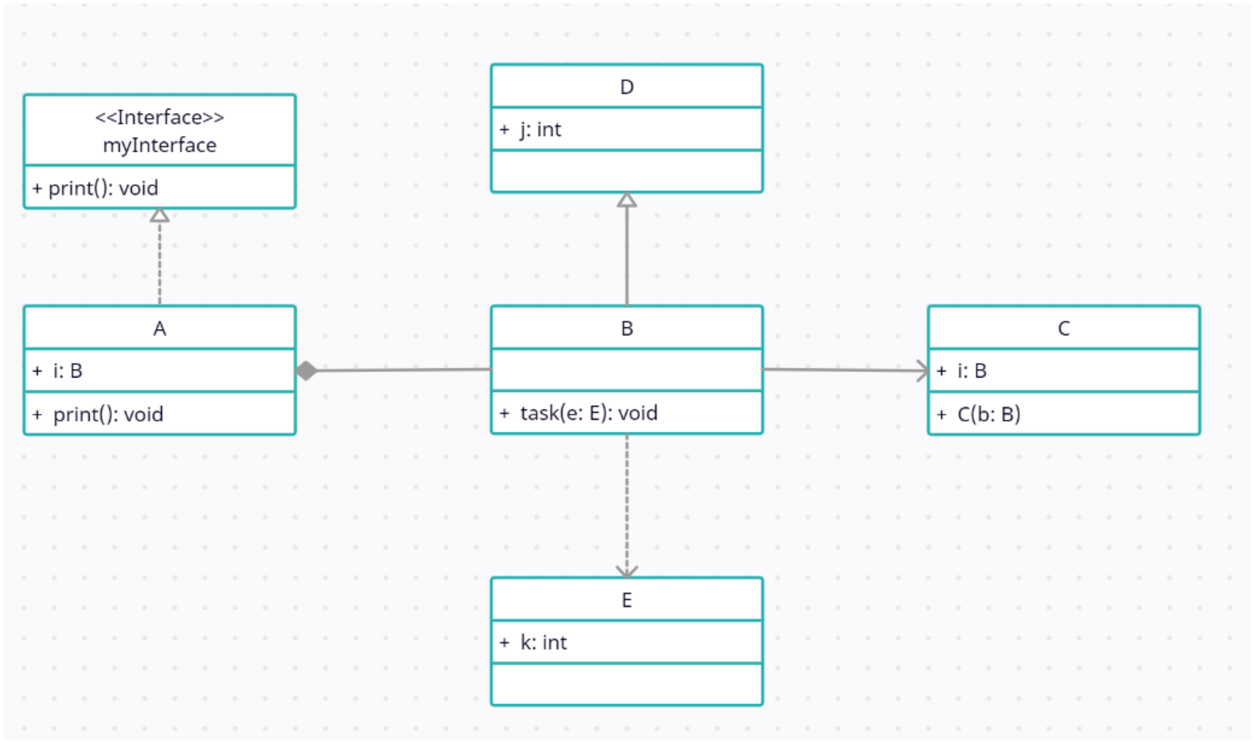
```
public class Main
{
    public static void main(String[] args) throws IOException {
        FileReader fr = new FileReader("Digits.txt");
        FileWriter fw = new FileWriter("DigitToWord.txt");
        int i;
        while ((i = fr.read()) != -1)
        {
            int c = i - '0';
            if (c == 0)
                fw.write("zero\n");
            else if (c == 1)
                fw.write("one\n");
            else if (c == 2)
                fw.write("two\n");
            else if (c == 3)
                fw.write("three\n");
            else if (c == 4)
                fw.write("four\n");
            else if (c == 5)
                fw.write("five\n");
            else if (c == 6)
                fw.write("six\n");
            else if (c == 7)
                fw.write("seven\n");
            else if (c == 8)
                fw.write("eight\n");
            else
                fw.write("nine\n");
        }
        fr.close();
        fw.close();
    }
}
```

+0.5	for throwing/handling suitable exceptions
+0.25 * 2	for reading files correctly (any I/O stream can be used)
+0.5	for reading file character by character
+1	for converting digits into spellings (deduct 0.25 if one case is missing (like zero))
	(deduct 0.5 if more than 1 case is missing)
+0.5	for adding nextline after each spelling
+0.5 * 2	for closing the files

Q4) Make a UML diagram for the following code [5 marks]

```
interface myInterface{
        public void print();
}
class A implements myInterface{
        B i = new B();
        public void print(){ System.out.println("Print done");  }
}
class B extends D{
        void task(E e){ System.out.println("Task over"); }
}
class C{
        B i;
        C(B b){ i=b; }
}
class D{
        int j;
}
class E{
        int k;
}
```

Ans:



+1 for mentioning correct modifiers (every member is public, i.e '+' in this case)
(-0.25 if 1,2 are wrong. -0.5 if more than 2 are wrong. 0 if not mentioned)

+0.5 for separating attributes and methods

+1.5 for writing attributes and methods properly (-0.25 if 1,2 are wrong. -0.5 if 3,4 are wrong. -1 if more than 4 are wrong)
+2 for class relationships (-0.5 for every wrong arrow, don't hear arguments that 'only arrowhead is slightly wrong' etc)

Q5) Create a CPU class which has a Processor, RAM and a price tag. Create Processor class and RAM class within the CPU class. Processor class comprises two instance variables: number of cores and manufacturer name. Similarly, RAM class comprises two instance variables: number of gigabytes and manufacturer name. CPU class also comprises of assemble function that takes a processor and a RAM and a price tag to develop a working CPU and give a price tag to it. In the main function, create instances of a CPU, Processor and RAM and assemble them. Then, display the working CPU using System.out.println (), which prints all the details of the working CPU. [6 marks]

Ans:

```
class CPU {
   double price;
   Processor p;
   RAM r;

   public String toString (){

   return p.toString()+r.toString()+price;
   }

   public void assemble (Processor p, RAM r, double price){

   this.price=price;
   this.p=p;
   this.r=r;
  }
   class Processor{

      double cores;
      String manufacturer;

      Processor(double cores, String manufacturer){
      this.cores=cores;
      this.manufacturer=manufacturer;
      }
      public String toString(){
         return manufacturer+cores;
      }
```

```
    }

    class RAM{

        double memory;
        String manufacturer;

        RAM(double memory, String manufacturer){
        this.memory=memory;
        this.manufacturer=manufacturer;
        }
        public String toString(){
            return manufacturer+memory;
        }
    }
}

class Main {
    public static void main(String[] args) {

        CPU cpu = new CPU();                                        // line 1

        CPU.Processor processor = cpu.new Processor(4,"aaa");       // line 2

        CPU.RAM ram = cpu.new RAM(4,"bbb");                         // line 3
        cpu.assemble(processor,ram,500);                            // line 4
        System.out.println(cpu);                                    // line 5

    }
}
```

+1	for assemble function
+0.5 * 3	for toString() methods (Give marks if student has handled printing without toString())
+0.5 * 2	for constructors of RAM and Processor
+0.5 * 5	for line 1,2,3,4,5 respectively