Java Core Assessment

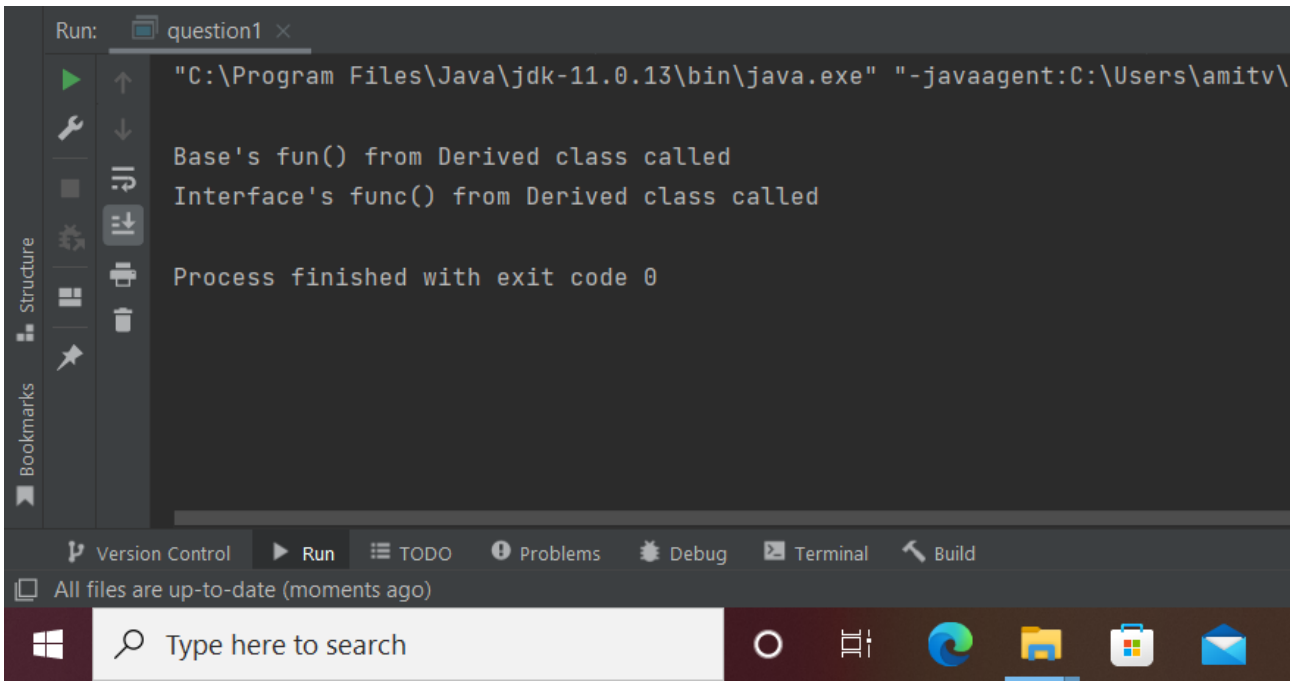
1. Implement your own abstract classes and interfaces.

Code : -

```
// Implement your own abstract classes and interfaces

package com.question1;

// Interface - Here methods are declared, but they should be called in class only by
overriding
interface Parent {
    void func();
}

// Class 1 - Abstract class with at least one abstract method
abstract class Base {
    abstract void fun();
}

// Class 2 - Normal class that extends abstract class for function overriding
class Derived extends Base implements Parent {
    void fun(){
        System.out.println("Base's fun() from Derived class called");
    }
    public void func(){
        System.out.println("Interface's func() from Derived class called");
    }
}

// Class 3 - Main class
public class question1 {
    public static void main(String[] args)
    {
        // Uncommenting the following line will cause compiler error as the line tries to
create an instance of abstract class.
        // Base b = new Base();

        // We can have references of Base type.
        Base b = new Derived();
        Parent p = new Derived();
```

```
        System.out.println();
        b.fun();
        p.func();
    }
}
```

Output : -



2. Implement your own encapsulating classes.

Code : -

```
// Implement your own encapsulating classes

package com.question2;

class EncapsulationDemo {
   // private members so that they can only be called by methods within class
   private int num1;
   private int num2;

   //getters and setters
   public int getNum1() { return num1; }
   public void setNum1(int num1) { this.num1 = num1; }
   public int getNum2() { return num2; }
   public void setNum2(int num2) { this.num2 = num2; }
}

public class question2 {
   public static void main(String[] args){
      EncapsulationDemo ed = new EncapsulationDemo();
      ed.setNum1(5);
      ed.setNum2(4);
      System.out.println("\nMultiplication of two numbers is " + ed.getNum1() *
ed.getNum2());
   }
}
```

Output : -

```
Run:  question2
"C:\Program Files\Java\jdk-11.0.13\bin\java.exe" "-javaagent:C:\Users\amitv\

Multiplication of two numbers is 20

Process finished with exit code 0
```

3. Create a thread named fetch Data using thread extend method

Fetch data should implement sleep method with 5000ms time

Create a thread named processData using runnable interface

Make sure processData starts its execution only after fetchData

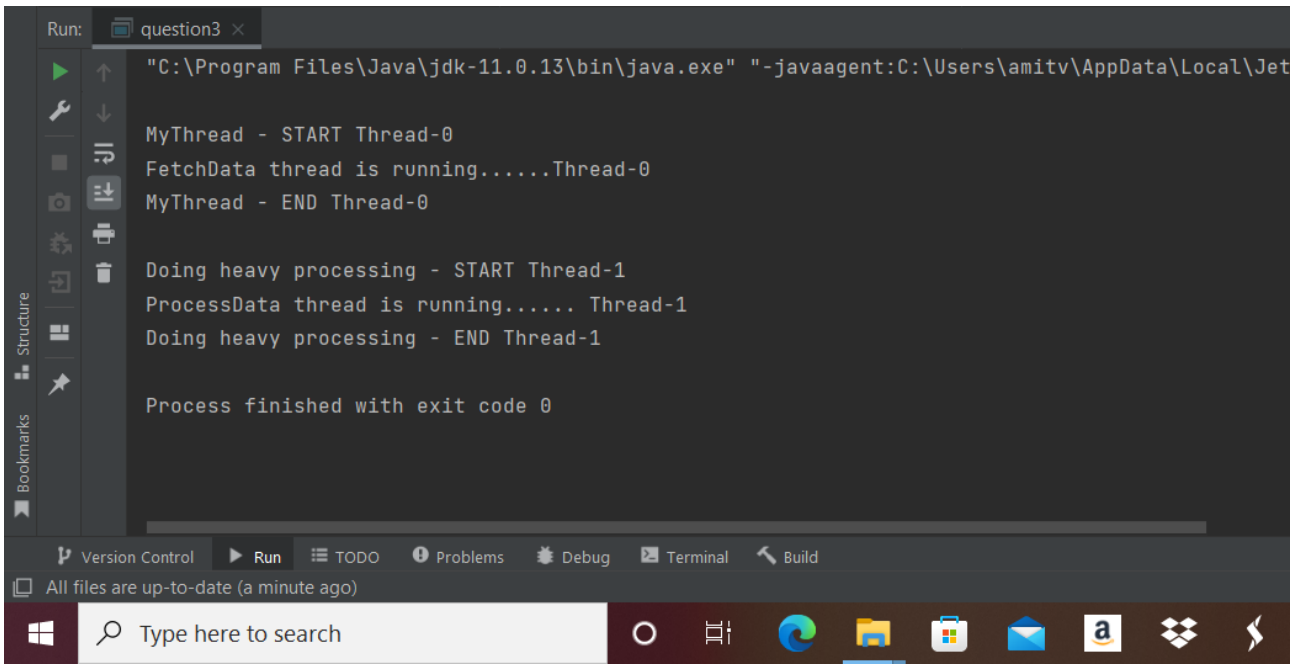
thread has completed its execution with the timeout of 10000ms

Code : -

```
// Create a thread named  fetch Data using thread extend method
//        Fetch data should  implement sleep method  with 5000ms time
//        Create a thread named  processData  using  runnable  interface
//        Make sure processData  starts its execution only after fetchData
//        thread has  completed its execution with the timeout of 10000ms

package com.question3;

class FetchData extends Thread{
    @Override
    public void run() {
        System.out.println("\nMyThread - START
"+Thread.currentThread().getName());
        try {
            Thread.sleep(5000);
            System.out.println("FetchData thread is running....." +
Thread.currentThread().getName());
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println("MyThread - END "+Thread.currentThread().getName());
    }
}

class ProcessData implements Runnable{
    @Override
    public void run() {
        System.out.println("\nDoing heavy processing - START " +
Thread.currentThread().getName());
            try {
                Thread.sleep(10000);
                System.out.println("ProcessData thread is running..... " +
```

```
Thread.currentThread().getName());
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        System.out.println("Doing heavy processing - END " +
Thread.currentThread().getName());
    }
}

public class question3 {
    public static void main(String[] args)throws InterruptedException{
        FetchData fd = new FetchData();
        Thread t1 = new Thread(new ProcessData(), "Thread-1");
        fd.start();
        fd.join();
        t1.start();
        t1.join();
    }
}
```

Output : -



4. Create a resource called message

○Message will have text as the field and isEmpty as the condition

○It has two synchronized functions read and write

●Create a writer thread that writes resource

●Create a reader thread that reads resource

Code : -

```
//Create a resource called message
// ○Message will have text as the field and isEmpty as the condition
// ○It has two synchronized functions read and write
// ●Create a writer thread that writes resource
// ●Create a reader thread that reads resource

package com.question4;

class Message {
    private String data;
    synchronized void writeData(String str) {//synchronized method
        this.data = str;
    }
    synchronized void readData() {//synchronized method
        System.out.println(data);
    }
}

class Writer extends Thread{
    Message t;
    Writer(Message t){
        this.t=t;
    }
    public void run(){
        t.writeData("\nHello World");
    }
}

class Reader extends Thread{
    Message t;
    Reader(Message t){
        this.t=t;
```

```
    }
    public void run(){
        t.readData();
    }
}

public class question4 {
    public static void main(String[] args){
        Message obj = new Message();
        Writer t1=new Writer(obj);
        Reader t2=new Reader(obj);
        t1.start();
        t2.start();
    }
}
```

Output : -