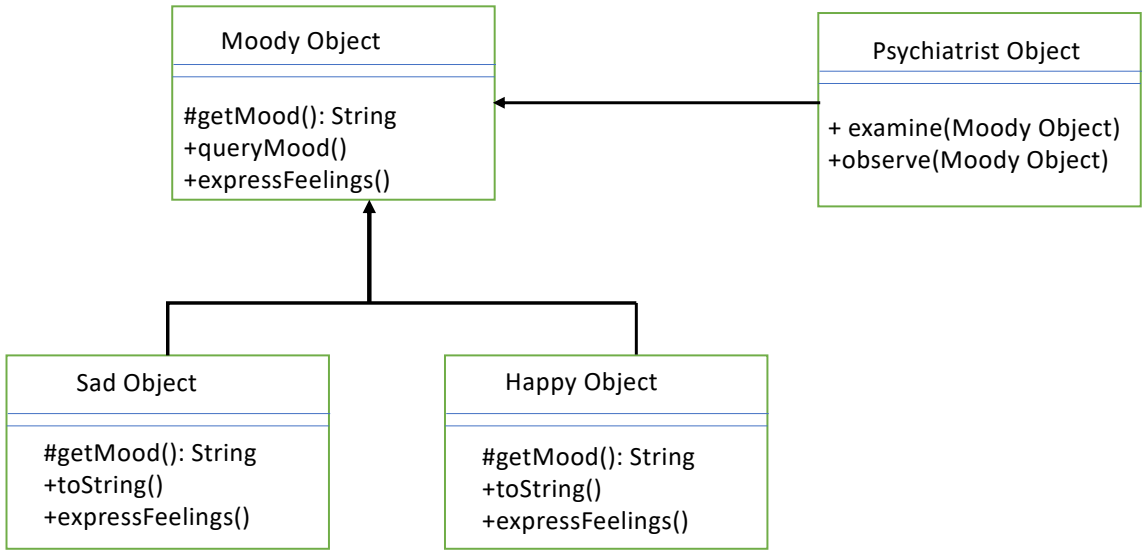
Q1. Write java program uses abstract classes and methods, the program should implement the design indicated in the UML diagram below.



Moody object :

```
// Return the mood : sad or happy – depending on which object sends the message
 abstract String getMpod();

// Each Object expresses a different motion
abstract void ExpressFeelings();

//an object responds according to how it feels, print ”I feel Happy(or Sad) today!!”
Public void queryMood()
```

Sad Object:

```
//returns string indicating sad
public String getMood();

//print crying string “‘waah’  ‘boo hoo’  ‘weep’ ‘sob’”
public void expressFeelings();

//returns message about self : “Subject cries a lot”
public String toString();
```

Happy Object:

```
//returns string indicating happy
public String getMood();

//print laughter string "heeehee….hahahah…HAHAHA!!"
public void expressFeelings();

//returns message about self: "Subject laughs a lot"
public String toString();
```

Psychiatrist Object:

```
//asks moody object about its mood
public void examine(moodyObject moodyObject);

//a moodyObject is observed to either laugh or cry
public void observe(moodyObject moodyObject);

//returns message about self: "Subject laughs a lot"
public String toString();
```

Write a main() method that creates a psychiatrist object and 2 moody objects. The psychiatrist object will examine and observe each moodyObject. Output of the program will look like below:

O/P:

How are you feeling today?

I feel happy Today

heeehee….hahahah…HAHAHA!!

Observation: Subject laughs a lot

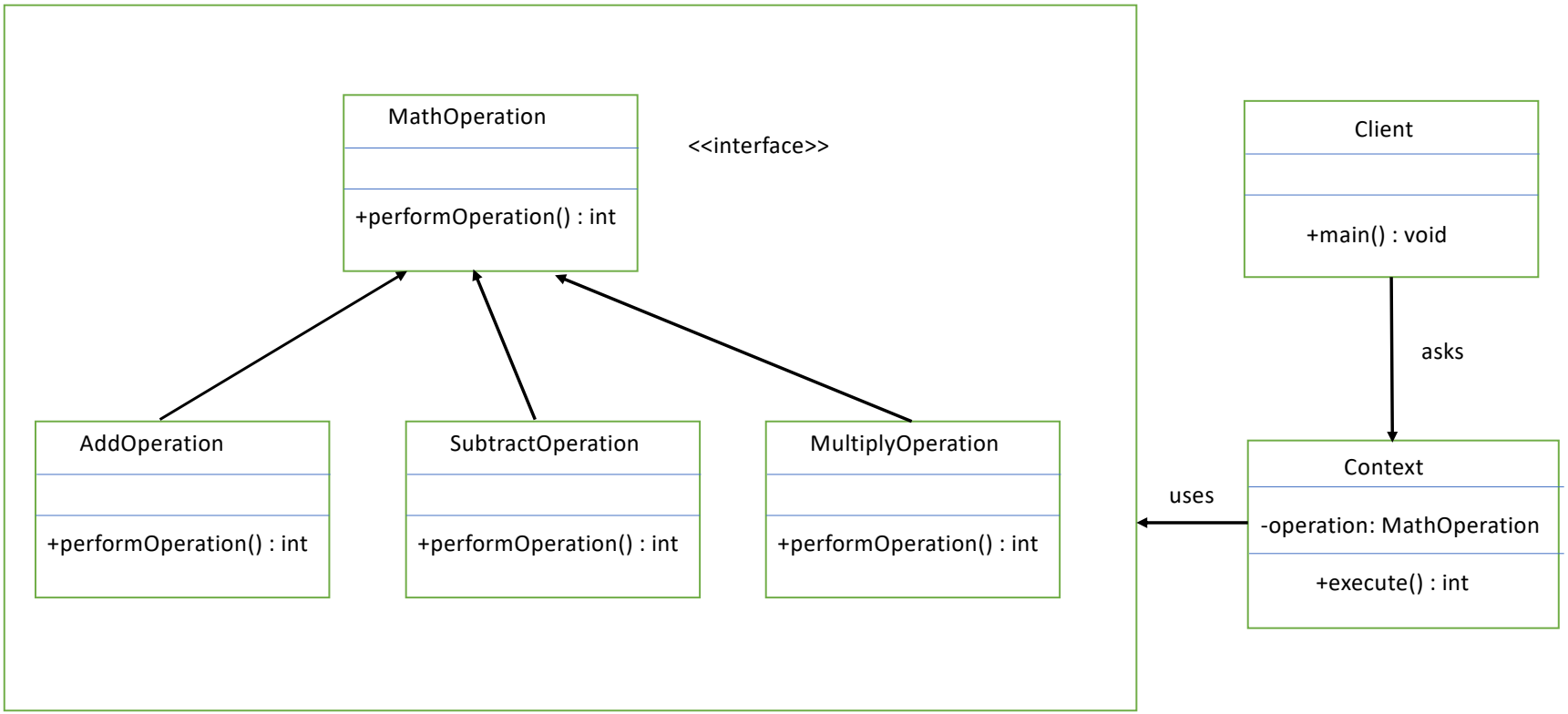
How are you feeling today?

I feel sad Today

'waah' 'boo hoo' 'weep' 'sob '

Observation: Subject cries a lot

Q2. This java program uses interfaces, the program should implement the design indicated in the UML diagram below.



MathOperation.java

```
public interface MathOperation {
    public int performOperation(int num1, int num2);
}
```

AddOperation.java

```
public class AddOperation implements MathOperation{
        // Implement your code here
}
```

SubtractOperation.java

```
public class SubtractOperation implements MathOperation{
        // Implement your code here
}
```

MultiplyOperation.java

```
public class MultiplyOperation implements MathOperation{
        // Implement your code here
}
```

Context.java

```
public class Context {
    MathOperation operation;
     public Context(MathOperation operation) {
          this.operation = operation;
    }
    public int execute(int num1, int num2) {
                // Implement your code here
    }
}
```

Client.java

```
public class Client {
     public static void main(String[] args) {
          Context contextAdd = new Context(new AddOperation());
          System.out.println(contextAdd.execute(5, 15)); // Expects 20
          Context contextSubtract = new Context(new SubtractOperation());
          System.out.println(contextSubtract.execute(50, 40)); // Expects 10
         Context contextMultiply = new Context(new MultiplyOperation());
          System.out.println(contextMultiply.execute(4, 25)); // Expects 100
     }
}
```

(Refer next page for instructions)

Instructions:

1. Write separate java files for each class or interface

2. Wherever you find "Implement your code here", implement your logic.

3. Do not modify any method signature.

4. Do not modify Client.java. We test your code by running the Client.java and it should give the expected output mentioned in the comments in Client.java