ABHISHEK DHAR
23201
E-10

ASSIGNMENT - II (OOP)

Title → Strategy Design Pattern

Aim → Implement and Apply Strategy Design Pattern for simple Shopping Cart where three payment strategies are used such as Credit Card, PayPal, BitCoin. Create the interface for strategy pattern and give concerete implementation for payment..

Objectives → To learn the concept of strategy design pattern.

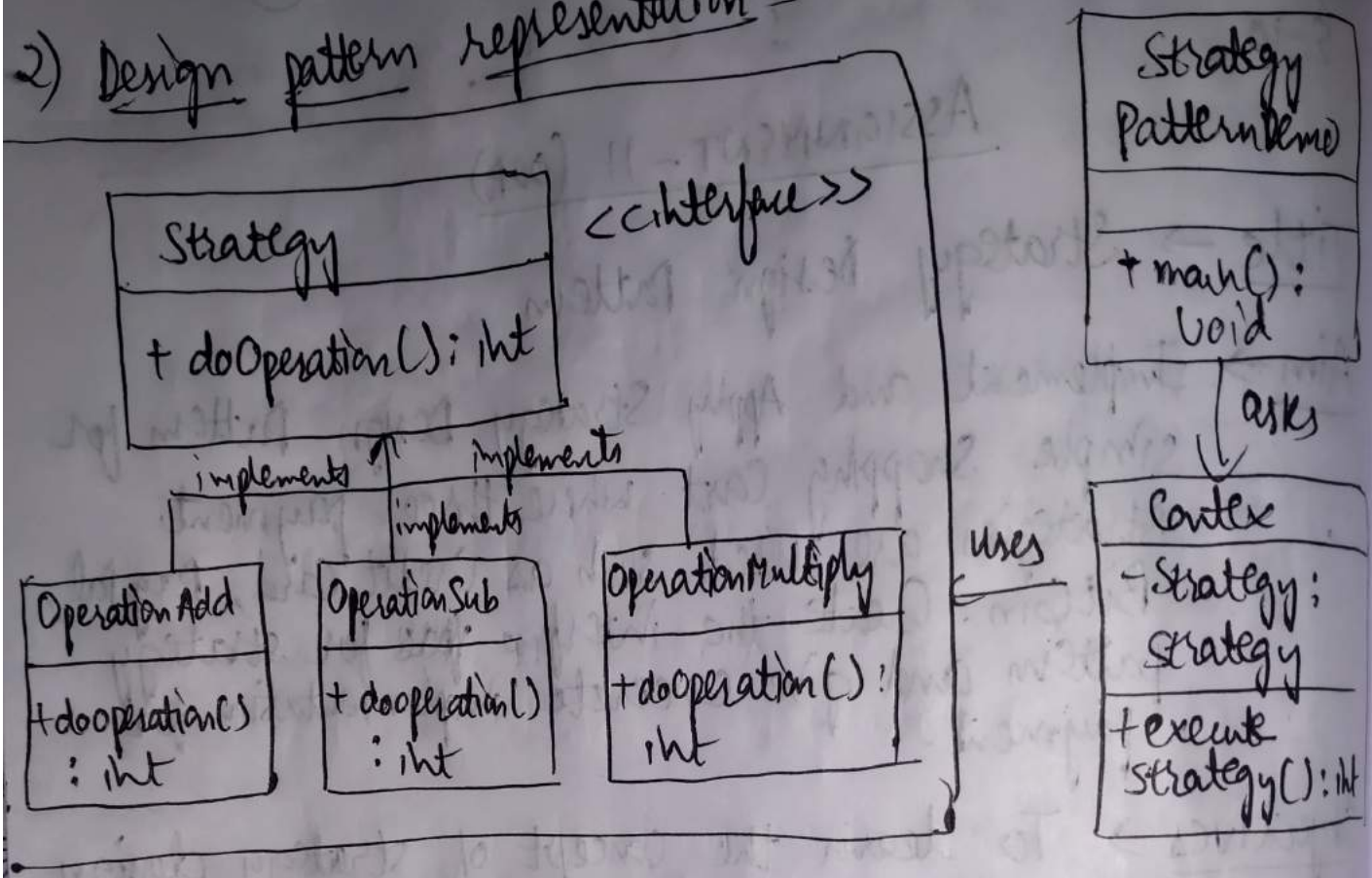
Theory →

1) What is strategy design Pattern?

In strategy design pattern a class behavior or its algorithm can be changed at runtime. This type of design comes under behavioral pattern. In strategy pattern, we create objects which represents various strategies and a context object whose behaviour varies as per its strategy object. The strategy object changes the executing algorithm of the context object. We are going to create a strategy interface defining an action and Concrete strategy classes implementing the strategy interface.

23201

2) Design pattern representation →



* Example →

Step 1 → Create an interface

Strategy.java

```
public interface Strategy
{ public int doOperation (int num1, int num2);
```

Step 2 → Create concrete classes implementing same interface

Operation Add.java

```
public class OperationAdd implements Strategy {
@Override
public int doOperation(int num1, int num2)
  return num1 + num2; }
```

Operation Subtract.java

```
public class Operation implements Strategy {
@Override
 public int doOperation(int num1, int num2) {
 return num1 - num2; }
}
```

Operation Multiply.java

```
public class OperationMultiply represents Strategy {
@Override
public int do Operation (int num1, int num2) {
return num1 * num2; }
}
```

Step3⇒ Create Context Class

Context.java

```
public class Context {
 private Strategy strategy;
 public Context (Strategy strategy) {
 this.strategy = strategy;
 }
public int executeStrategy (int num1, int num2)
{
   return strategy.doOperation (num1, num2); }
}
```

Step 4 → Use the context to see change in behaviour when changes its strategy.

```
Strategy Pattern Demo.java
public class Strategy Demo.java
public static void main (String [] args) {
 Context context = new Context (new Operation Add());
System.out.println("10+5=" + context.execute strategy
                                        (10,5));
Context = new Context (new Operation Subtract ());
System.out.println("10-5=" + context.execute strategy
                                        (10,5));
Context = new Context (new Operation Multiply());
System.out.println("10x5=" + context.execute strategy
                                        (10,5));
}
}
```

Output →

```
10+5 = 15
10-5 = 5
10*5 = 50
```

Output →

```
Strategy Pattern -
100 paid using PayPal
100 paid using Credit/debit card
120 paid using PayPal
120 paid using Credit/debit Card
```

23201

* Advantages →

1) A family of algorithms can be defined as a class hierarchy and can be used be interchangeably to alter application behaviour without changing its architecture.
2) By encapsulating the algorithm separately, new algorithms complying with the same interface can be easily introduced.

* Disadvantages →

1) The application must be beaware of all strategies to select right one for right situation.
2) In most cases, the application configures the context with the reqd. strategy object. Therfore, the application needs to create & maintain two objects in place of one.

* Applications →

1) Save file in different formats.
2) Run various sorting algorithms
3) File compression.