






ENSF 409 - Lab...



Exercise - 2: SOLID Principles (35 Marks)

Task 1) (10 marks) Consider the following classes in answering parts a and b.

Part a (3 marks) Which SOLID principle is this program violating? Briefly explain.

risk of Substitution since the class Vehicle is not general enough so its child can replace it. So I added 2 more classes that inherits Vehicle so their children can be more relatable when extend them.

Part b (7 marks) If needed, re-write each class completely to remove the code smell in question. If not, simply write no change needed for a class.




Write any classes or interfaces you are adding to this program in this space:

```
abstract public class EnginedVehicle extends Vehicle {  
    EnginedVehicle (String vType, String eType) {  
        super (vType, eType);  
    }  
}  
abstract public class Non-enginedVehicle extends Vehicle {  
    Non-enginedVehicle (String vType) {  
        super (vType, "null");  
    }  
}
```

```
abstract public class Vehicle {  
    abstract public String getVehicleType ();  
    abstract public String getEngineType ();  
}
```

```
//If this class needs to change, rewrite it  
//completely. If not, write: "No change needed".  
abstract public class Vehicle {  
    private String vehicleType;  
    private String engineType;  
    Vehicle (String vType, String eType) {  
        vehicleType = vType; engineType = eType;  
    }  
    public String getVehicleType() return vehicleType;  
    public String getVehicle Engine() return engineType;  
}
```

Page 4 of 5



```
public class Car extends Vehicle{  
  
    private String vehicleType;  
    private String engineType;  
    Car(String vType, String eType) {vehicleType  
        = vType; engineType = eType;  
    }  
    @Override  
    public String getVehicleType() {  
        return vehicleType;  
    }  
    @Override  
    public String getEngineType() {  
        return engineType;  
    }  
}
```

```
//If this class needs to change, rewrite it  
//completely. If not, write: "No change needed".  
public class Car extends EnginedVehicle {  
  
    Car (String vType, String eType) {  
        super (vType, eType);  
    }  
}
```

```
public class Bicycle extends Vehicle{  
  
    private String vehicleType;  
    Bicycle(String vType) {  
        vehicleType = vType;  
    }  
    @Override  
    public String getVehicleType() {  
        return vehicleType;  
    }  
  
    @Override  
    public String getEngineType() {  
        return null;  
    }  
}
```

```
//If this class needs to change, rewrite it  
//completely. If not, write: "No change needed".  
public class Bicycle extends Non-enginedVehicle {  
    private String vehicleType  
    Bicycle (String vType) {  
        super (vType);  
    }  
}
```

Task 2) (15 marks) Consider the following classes in answering parts a, b, and c.




```
public class GraphicCreator {  
  
    public void drawShape(Shape s) {  
        if (s.getShapeType() == 1)  
            drawSquare((Square) s);  
        else if (s.getShapeType() == 2)  
            drawCircle((Circle) s);  
    }  
    public void drawCircle(Circle c) { //Some code. }  
    public void drawSquare(Square s) { //Some code. }  
}
```

```
public class Shape {  
  
    private int shapeType;  
  
    Shape (int type){shapeType = type;}  
    public int getShapeType() {  
        return shapeType;  
    }  
}
```

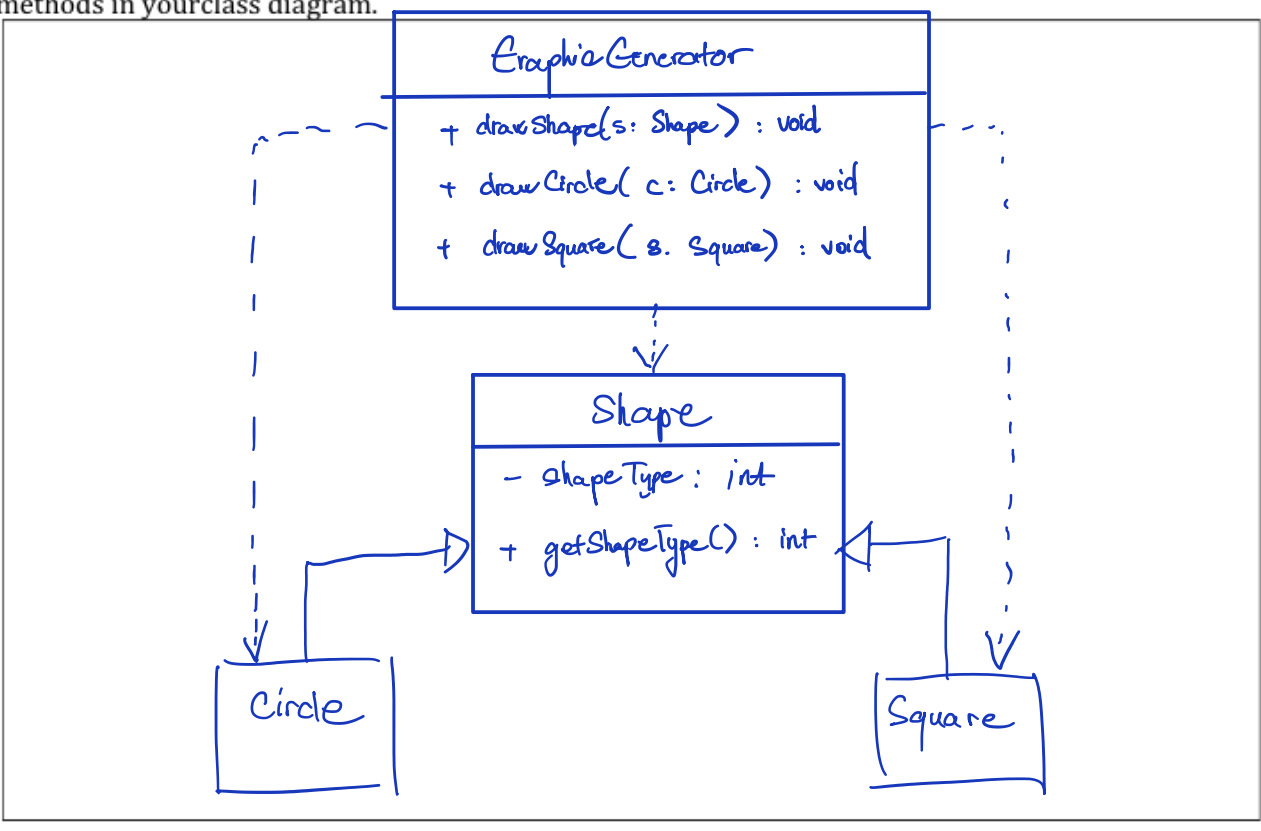
```
public class Circle extends Shape(Circle  
    (){super(2);}
```

```
public class Square extends Shape(Square  
    (){super(1);}
```

Page 5 of 5






Part a (5 marks) Draw the class diagram for the classes above. Make sure to include all fields and methods in your class diagram.



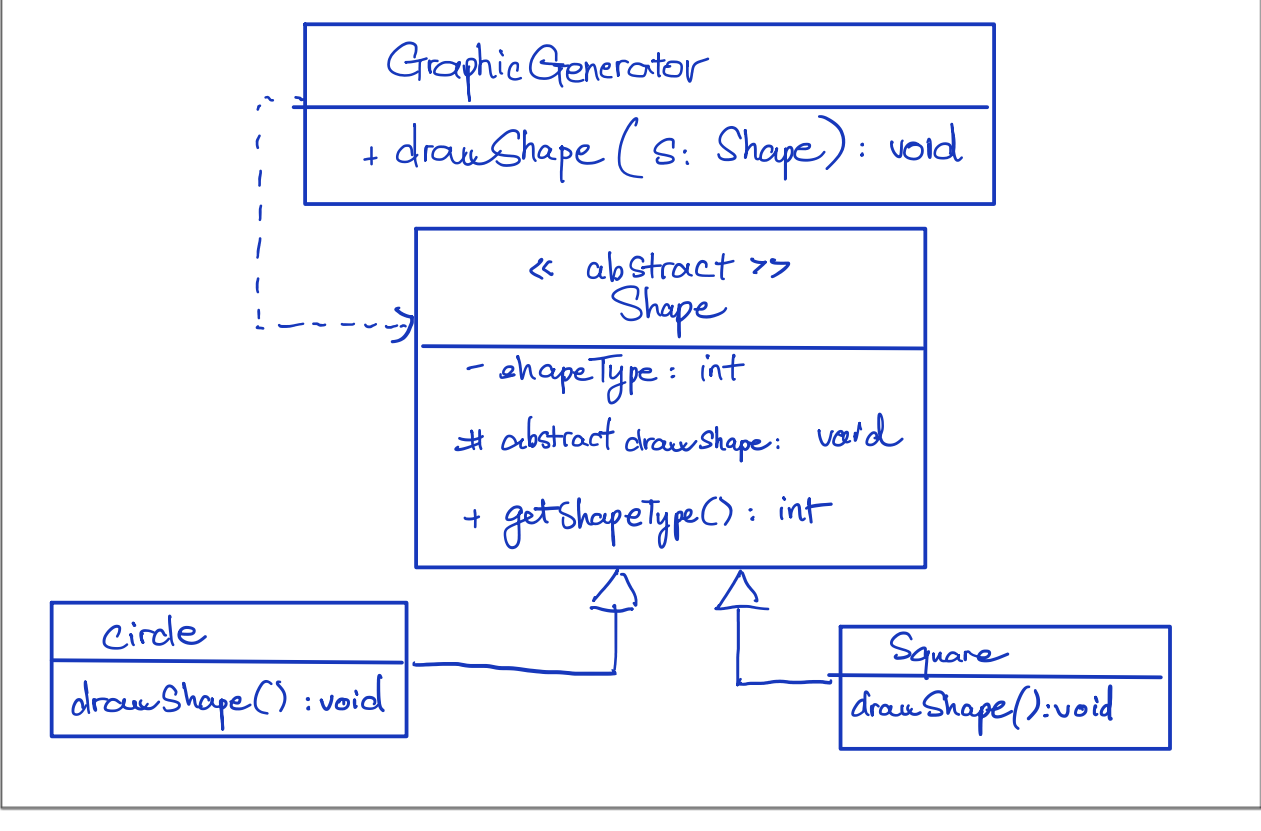
Part b (4 marks) Which SOLID principle is this program violating? Briefly explain.

This program violates Open/Closed Principle .
If we try to add more classes for different shapes, direct modifications on the class GraphicGenerator is inevitable; which may causes unexpected bugs.

Page 6 of 5



Part c (6 marks) Draw a class diagram for a proper design that removes the code smell in the above program. Clearly include all fields and methods in your class diagram.



What to hand in: Please submit your solutions in a pdf file for tasks 1 and 2.

Page 7 of 5