

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.

The program violates the Liskov Substitution Principle, which states that every subclass should be able to be substituted for its super/parent class. This breaks here b/c bikes don't have engines.

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

```
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 {
    abstract public String getVehicleType();
}
```



```java
public class Car extends Vehicle{
```
MotorVehicle {

```java
	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;
	}
}
```

```java
//If this class needs to change, rewrite it
//completely. If not, write: "No change needed".
```

Sorry, no time, only changed one thing

Change superclass from Vehicle to MotorVehicle.

```java
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;
	}
}
```

Same, but removed this chunk

```java
//If this class needs to change, rewrite it
//completely. If not, write: "No change needed".
```



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

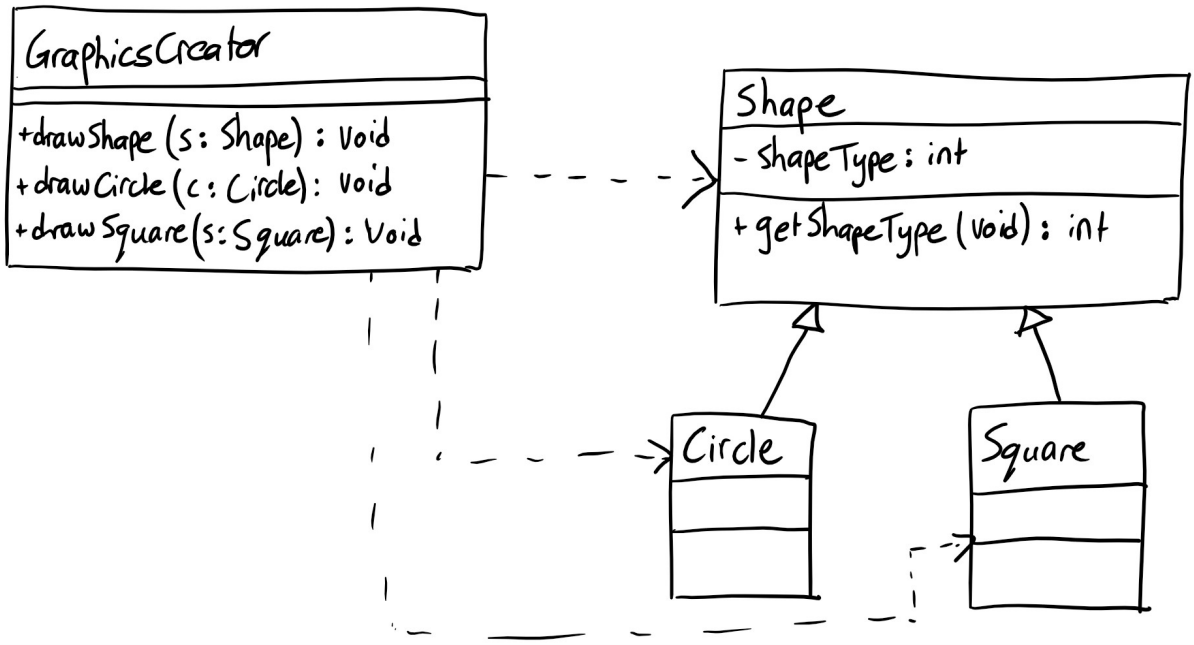
```
public class GraphicCreator {
	public void drawShape(Shape s) {
		if (s.getShapeType() == 1)
			drawSquare((Square) s);
		else if (s.getShapeType() == 2)
			drawCircle((Circle) s);
	}
	public void drawCircle(Circle c) {//Some code. }
	public void drawSquare(Square s) {//Some code. }
 }
```

```
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);}
}
```

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

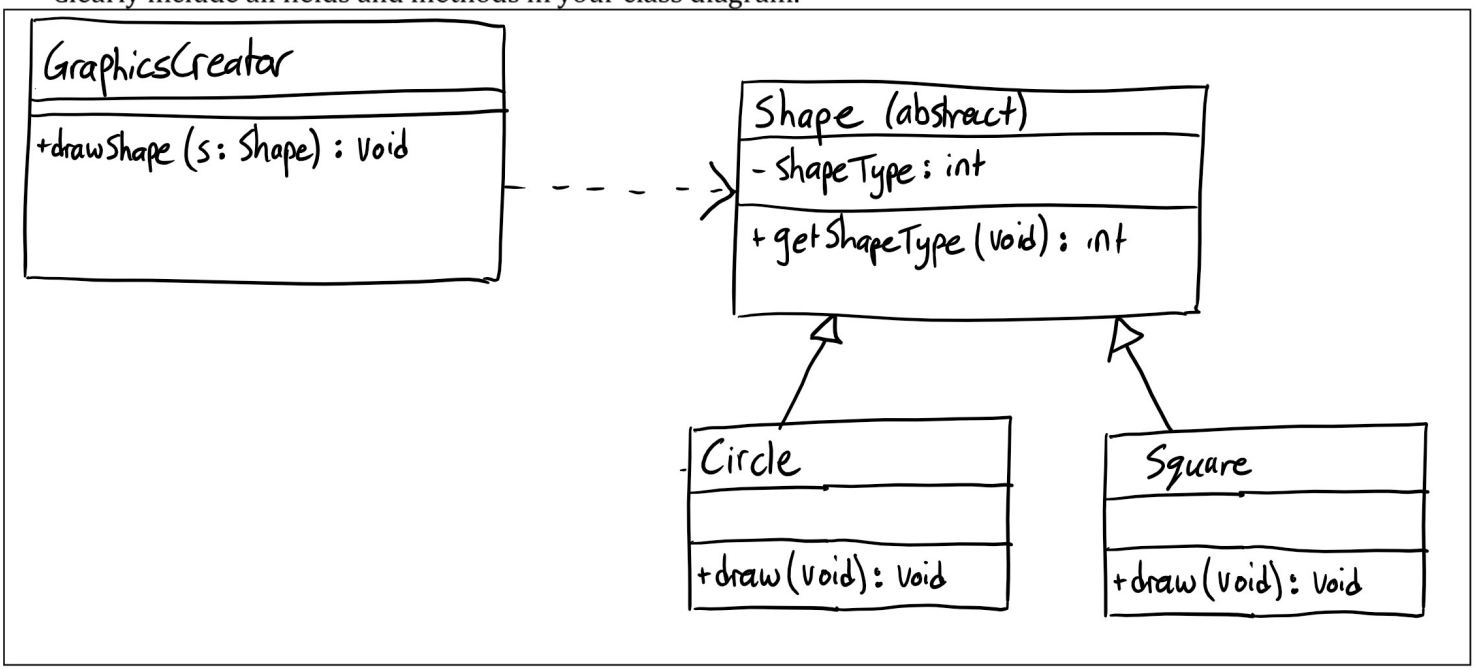


Part b (4 marks) Which SOLID principle is this program violating? Briefly explain.

Violates the open-close principle. This principle focuses on allowing extention by adding to subclasses, instead of modifying the super/abstracted class. In this instance, each shape should have its own draw() method, instead of having different draw methods for each one in the GraphicCreator. The drawShape method can then just call the Shape's draw method. If another shape is implemented, GraphicsCreator would require significant modification.



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.

How to submit: Submit your PDFs from Exercises 1 – 3 to D2L.