CSC 3380 Enterprise Architect (EA) Homework #2

Aymond, CSC 3380 Section 1, Louisiana State University Due 11PM 3/20/2020

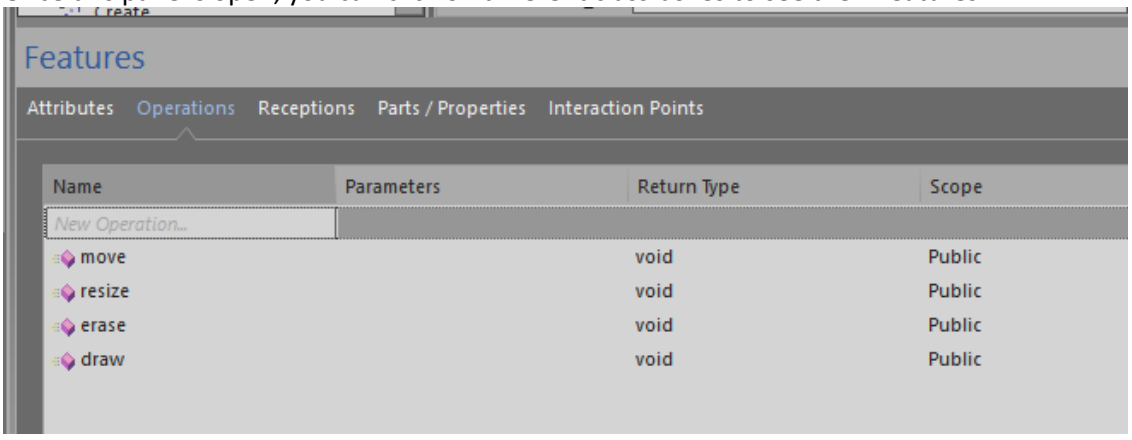
For this assignment, you will create a UML class diagram for the Java code skeleton provided below, using Enterprise Architect (EA).

I recommend the following steps in creating a class diagram:

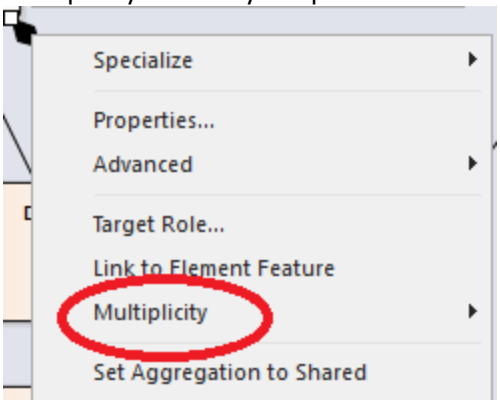
1. Create a class for each class and interface in your design.
2. Add the class details (e.g., member variables, methods), with appropriate visibility and details (e.g., data types, parameter datatypes).
3. Add appropriate connections between classes.
4. Rearrange classes so that their layout is easy to understand and eliminates (or minimizes) overlapping connections.

A few things worth noting:

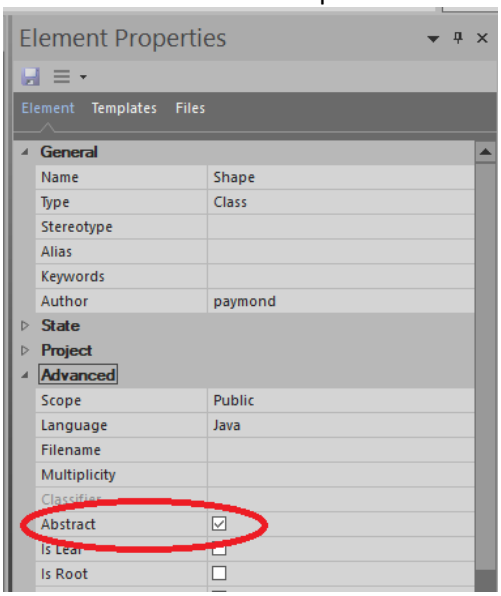
- In the UML diagram, abstract methods in an abstract class should not be repeated in the classes that override them.
- The implementation of an interface is modeled with the realization relationship in UML.
- When classes contain member variables that are classes themselves, rather than listing the member variable in the class with the autonomous member variables (e.g., int, float, long, String), they are designated in their own class box with the appropriate relationship connecting the two class boxes.
- To add methods (operations) and member variables (attributes) to a class in EA, right click on the class box and select Features/Operations or Features/Attributes. A Features frame will open at the bottom of the window. You can tab between Attributes and Operations to enter values. Once this pane is open, you can click on different class boxes to see their Features.



- To set the multiplicity of a relationship, right click on the end of the connection and select Multiplicity. A variety of options are available to you.



- To make a class an abstract class in EA, you can check the Abstract option in the Advanced section of the Element Properties for the class box



- Our diagrams are never complete until we have thanked Sparx Systems for allowing us to use EA. On all artifacts that you create from EA, you will need to add a Note (found in the Common section of the Toolbox) and include this message on the note: “Thanks to Sparx Systems for allowing LSU students and faculty use of the Enterprise Architect for academic purposes.”
- Convert your diagram to a jpg image. Zip and upload the image along with your eapx file to Moodle. The zip file should be named for your PawsID (your LSU email address before the @).

Java Code:

```
public class Frame {...}

public interface Event {...}




public class Window extends Frame implements Event{

	private DrawingContext drawingContext;
	private Shape shape[];	//1..* In this context, a Shape can exist without a
				//Window

	private void close() {...}
	private void display() {...}
	private void handleEvent() {...}
	private void move() {...}
	private void open() {...}

	...
}

public class DrawingContext {
	private void clearScreen() {...}
	private void getHorizontalSize() {...}
	private void getVerticalSize() {...}
	private void setPoint() {...}

	...
}

public class ConsoleWindow extends Window {...}

public class DialogBox extends Window {

	private DataControlBar dataControlBar;

	...
}

public class DataControlBar {...}

public abstract class Shape {

	abstract public void draw();
	abstract public void erase();
	abstract public void move();
	abstract public void resize();

}
```

```java
public class Circle extends Shape {

	private int center;
	private float radius;
	private Point point[];	//1..* In this context, a Point cannot exist without
				//a Circle

	public double area(float param) {...}
	public double circum() {...}
	public void setCenter(int param) {...}
	public void setRadius(float param) {...}

	@Override
	public void draw() {...}

	@Override
	public void erase() {...}

	@Override
	public void move() {...}

	@Override
	public void resize() {...}

}

public class Point {...}

public class Rectangle extends Shape {

	@Override
	public void draw() {...}

	@Override
	public void erase() {...}

	@Override
	public void move() {...}

	@Override
	public void resize() {...}
}
```

```
public class Polygon extends Shape {

	@Override
	public void draw() {...}

	@Override
	public void erase() {...}

	@Override
	public void move() {...}

	@Override
	public void resize() {...}
}
```