May 30th, 2022

CENG 431 Building Software Systems
Final Exam

Closed Book Exam. Answer the following questions in 75 minutes.

Questions:

1. **Draw the UML sequence diagram for performing a sale with respect to Figure 1 (15 pts). Fill the circles in Figure 1 with the order of implementation (5 pts) and explain why (5 pts). Write the Register and the Sale classes in Java (30 pts).**

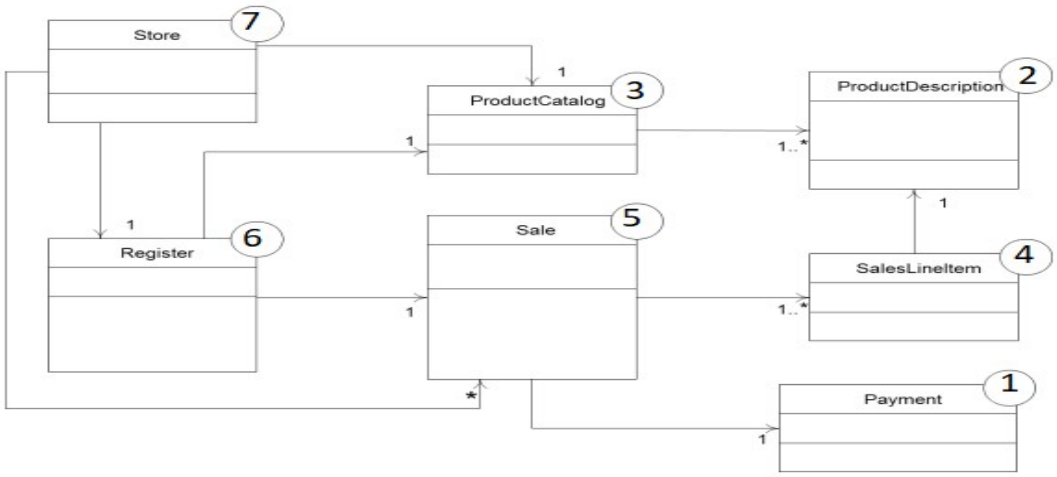
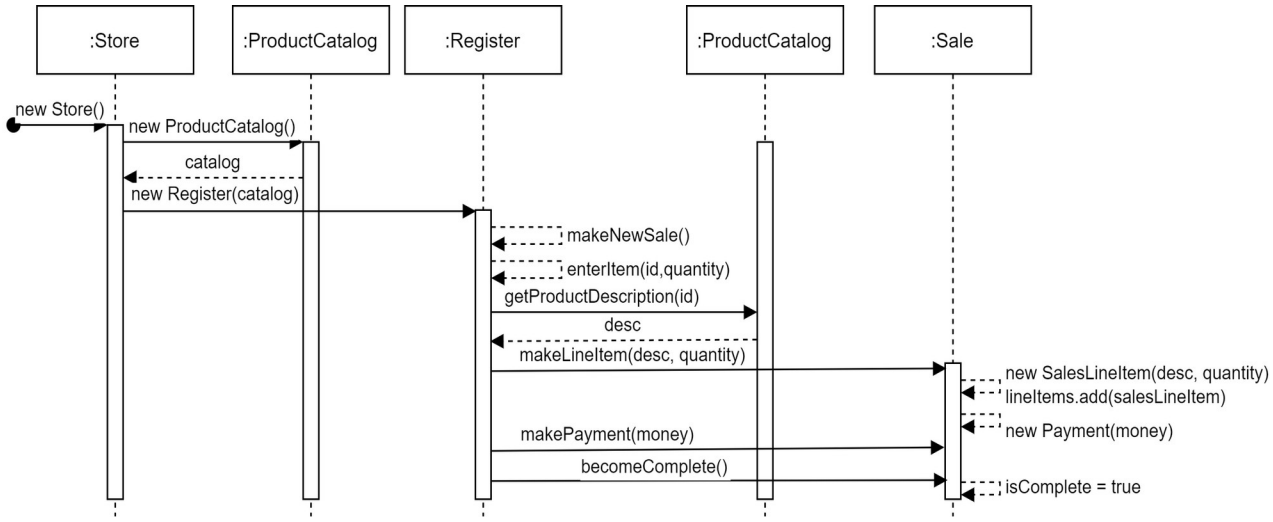


Figure 1. Class Diagram for Store Domain

Classes need to be implemented (and ideally, fully unit tested) from leastcoupled to most-coupled.



2. Explain implementation of Strategy Design Pattern with one example in Java (20 points).

A class defines many behaviors, and these appear as multiple conditional statements in its operations. Instead of many conditionals, move related conditional branches into their own Strategy class.

-Strategy (Compositor) declares an interface common to all supported algorithms. Context uses this interface to call the algorithm defined by a ConcreteStrategy.
-ConcreteStrategy implements the algorithm using the Strategy interface.
-Context (Composition) is configured with a ConcreteStrategy object, maintains a reference to a Strategy object, and may define an interface that lets Strategy access its data.

```
public interface InterestStrategy {
	public double interest(double principal, double rateOfInterest,
int timePeriod);
}
public class SimpleInterest implements InterestStrategy {
	@Override
	public double interest(double principal, double rateOfInterest,
int timePeriod) {
		return principal * rateOfInterest * timePeriod;
	}
}
public class CompoundInterest implements InterestStrategy {
	@Override
	public double interest(double principal, double rateOfInterest,
int timePeriod) {
		return principal * Math.pow((1 + rateOfInterest),
timePeriod) - principal;
	}
}
public class InterestCalculator {
	private InterestStrategy strategy;
	public InterestCalculator(InterestStrategy strategy) {
		this.strategy = strategy;
	}
	public double calculateInterest(double principal, double
rateOfInterest, int timePeriod) {
		return strategy.interest(principal, rateOfInterest,
timePeriod);
	}
}
public class Demo {
	public static void main(String[] args) {
		InterestCalculator calculator = new InterestCalculator(new
SimpleInterest());
		System.out.println("Simple Interest = " +
calculator.calculateInterest(200, 0.05, 7));
```

```
            calculator = new InterestCalculator(new
CompoundInterest());
            System.out.println("Compound Interest = " +
calculator.calculateInterest(200, 0.05, 7));
      }
}
```

3. **List SOLID principles (10 points) and give an example for implementation of the "L" principle in Java (15 points).**

Single Responsibility

Open/Closed

Liskov Substitution

Interface Segregation

Dependency Inversion

```
public class Square extends Rectangle {
      @Override
      public void setWidth(int width) {
            super.setWidth(width);
            super.setHeight(width);
      }

      @Override
      public void setHeight(int height) {
            super.setHeight(height);
            super.setWidth(height);
      }
}
```

Mathematically square is rectangle however, if we implement square as a rectangle, we violate Liskov Substitution Principle since their behaviors differ. Instead, we should implement rectangle and square as a shape:

```
public interface Shape {
    long area();
}

public class Square implements Shape {
      private int size;

      public Square(int size) {
            this.size = size;
      }

      @Override
      public long area() {
```

```
            return size * size;
      }

      public void setSize(int size) {
            this.size = size;
      }
}

public class Rectangle implements Shape {
      private int width;
      private int height;

      public Rectangle(int width, int height) {
            this.width = width;
            this.height = height;
      }

      @Override
      public long area() {
            return width * height;
      }

      public void setWidth(int width) {
            this.width = width;
      }

      public void setHeight(int height) {
            this.height = height;
      }
}
```