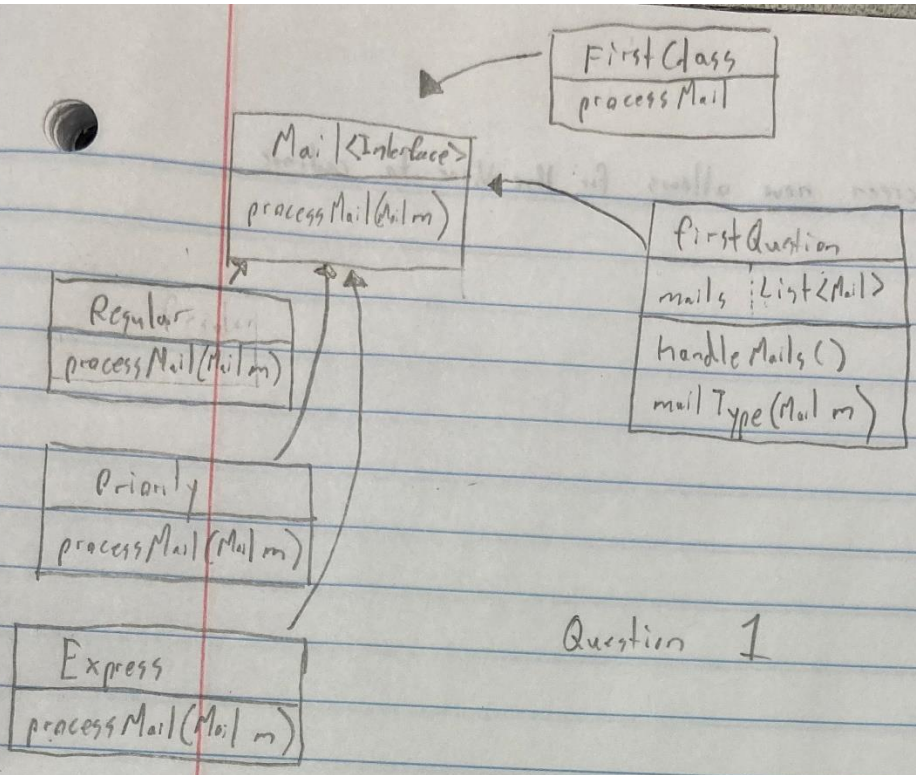
```java
import java.util.*;
class firstQuestion{

	private List<Mail> mails;

	// unimportant code omitted

	public void handleMails(){

		for (int i = 0; i < mails.size(); i++){

			mailType(mails.get(i));

		}

	}

	public void mailType(Mail m) {
		m.processMail(m);
	}

}

interface Mail{

	void processMail(Mail m);
}

class Regular implements Mail{

	@Override
	public void processMail(Mail m) {

	}

}

class Priority implements Mail{

	@Override
	public void processMail(Mail m) {

	}

}

class Express implements Mail{

	@Override
	public void processMail(Mail m) {

	}

}

class firstClass implements Mail{

	@Override
	public void processMail(Mail m) {
```

```
        }
}
```



Question 1

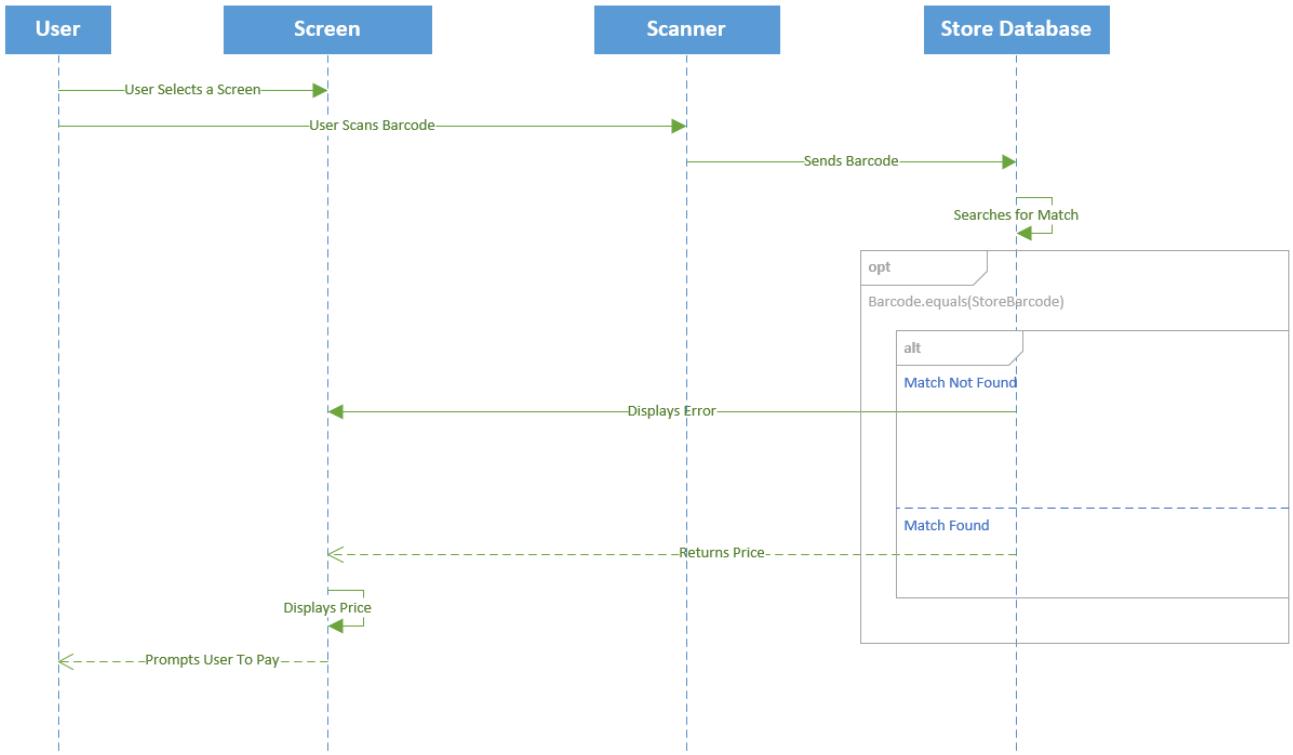
2) Precondition: Screen is already prepared to start
Assumption: PAF can detect what may go wrong

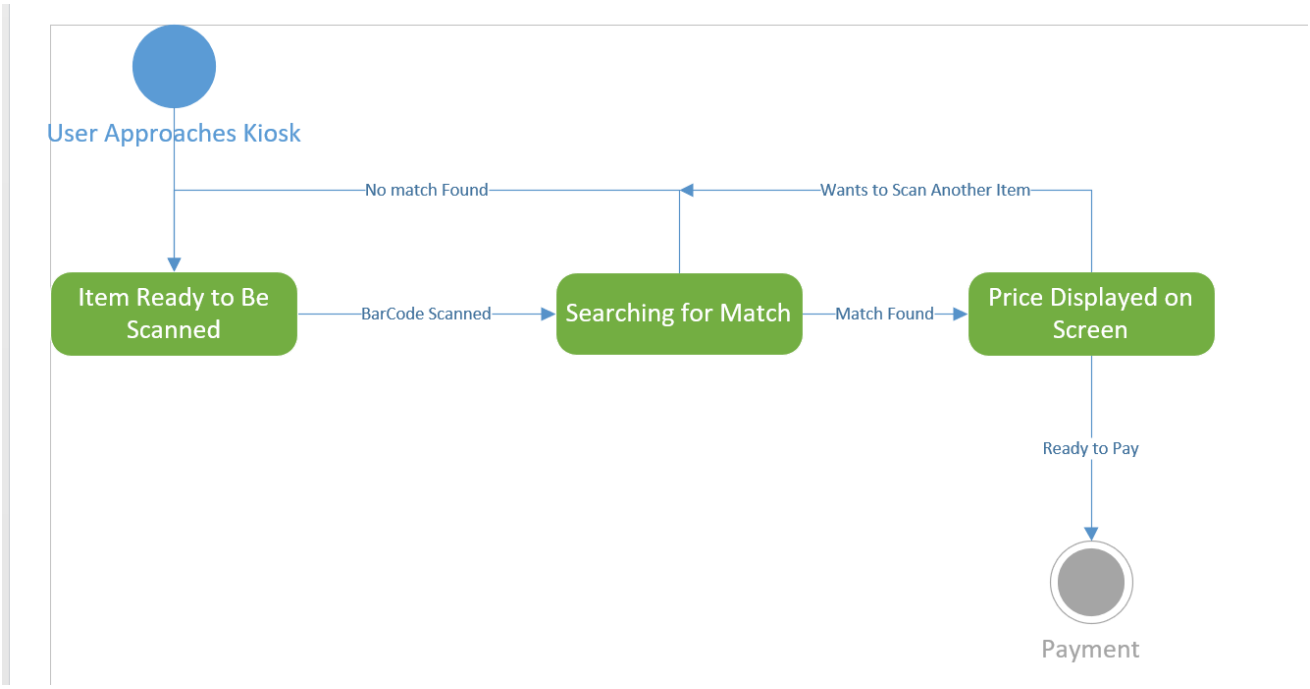
1. User selects a table to checkout at
2. PAF gets the barcode of the item scanned
3. PAF connects to the store's barcode database
4. PAF compares customer's barcode to stored barcodes.
5. PAF finds a match and returns the price value to the screen
6. Screen displays price of scanned item to user

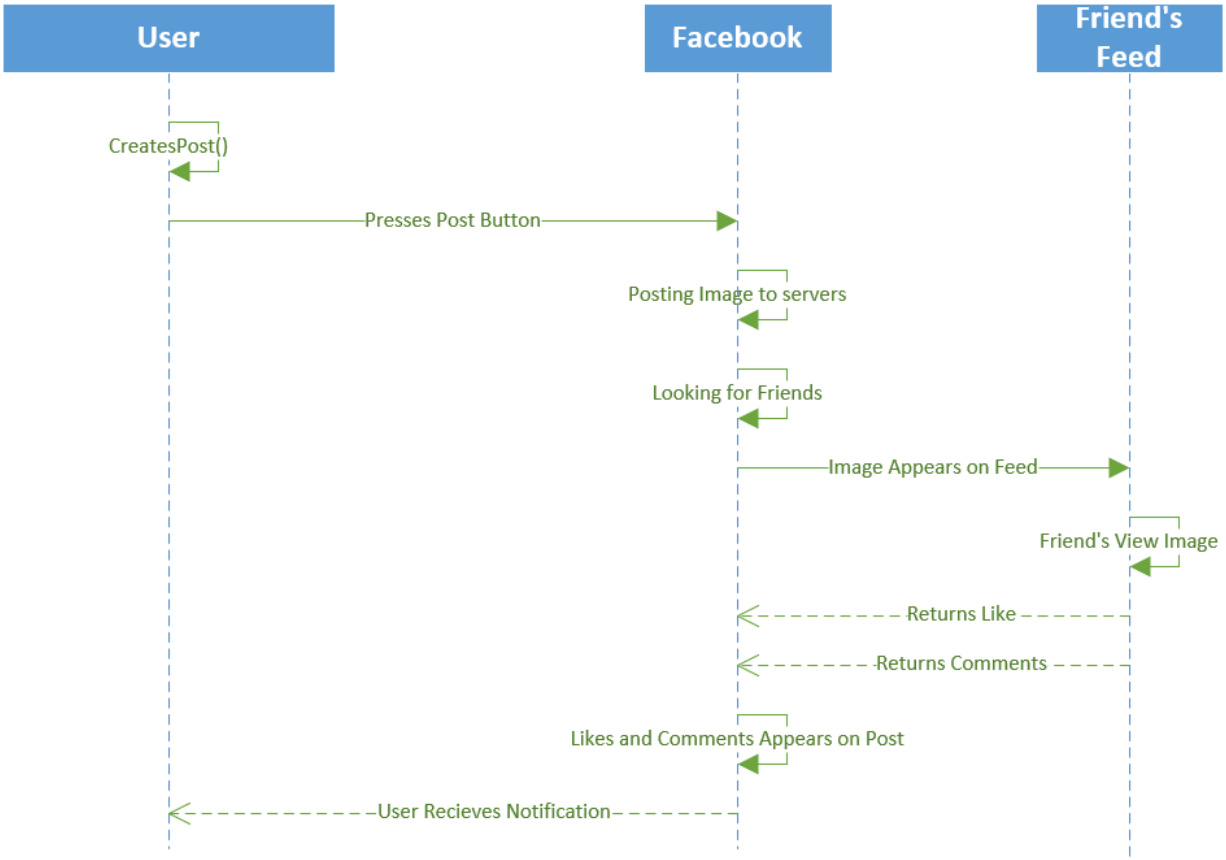
Extension

6a Item displayed is age restricted
6a.1 Screen displays message that the Item is age restricted
6a.2 User scans valid identification and gets barcode
6a.3 PAF connects to identification database
6a.4 PAF compares scanned barcode to other barcodes
6a.5 PAF finds a match and updates the screen

bab The screen now allows for the User to continue scanning


User
Screen
Scanner
Store Database
User Selects a Screen
User Scans Barcode
Sends Barcode
Searches for Match
opt
Barcode.equals(StoreBarcode)
alt
Match Not Found
Displays Error
Match Found
Returns Price
Displays Price
Prompts User To Pay


User Approaches Kiosk
No match Found
Wants to Scan Another Item
Item Ready to Be Scanned
BarCode Scanned
Searching for Match
Match Found
Price Displayed on Screen
Ready to Pay
Payment




```
import java.util.*;

interface TeamMember {
	void perform();
}

class Leader implements TeamMember{
	private String name;

	Leader(String name){
		this.name = name;
	}

	@Override
	public void perform() {
		//perform an action
	}

}

class Member implements TeamMember{
	private String name;

	Member(String name){
		this.name = name;
	}

	@Override
	public void perform() {
		//perform an action
```

```
	}

}

class Group implements TeamMember{

	List<TeamMember> team;
	@Override
	public void perform() {
		//perform an action
	}

	public void addMember(TeamMember member) {
		team.add(member);
	}

	public void removeMember(TeamMember member) {
		team.remove(member);
	}

}
```

```
interface IFunction {
	double function(int n);
}

class main{
	public double sum(IFunction fun, int x) {
		double answer = 0;

		while(x > 0) {
			x += fun.function(x);
		}
		return answer;
	}
}

class Cos implements IFunction{

	@Override
	public double function(int x) {
		double answer = Math.sin(x);
		return answer;
	}

}

class Sin implements IFunction{

	@Override
	public double function(int x) {
```

```
            double answer = Math.cos(x);
            return answer;
        }

        }
```

```
public class Light {

        private boolean bulb;

        Light(){
                off();
        }

        public void on(){
                bulb = true;
        }

        public void off(){
                bulb = false;
        }
}

class Controller{
        protected commandReciever command;

        public void setCommand(commandReciever c) {
                command = c;
        }

        public void controllerExecuter() {
                command.execute();
        }
}

interface commandReciever{
        void execute();
}

class turnOn implements commandReciever{

        Light light;

        turnOn(){
                light = new Light();
        }

        @Override
        public void execute() {
```

```
		light.on();
	}

}

class turnOff implements commandReciever{

	Light light;

	turnOff(){
		light = new Light();
	}

	@Override
	public void execute() {
		light.off();
	}

	}









public interface Lamp{

	public void turnOn();

	public void turnOff();
}

class Switch implements Lamp{
	protected boolean status;

	Switch(){
	}

	@Override
	public void turnOn() {
		status = true;
	}

	@Override
	public void turnOff() {
		status = false;
	}

	public String toString() {
		String str = "The status of the light bulb is ";

		if(status) {
			str += "on";
		}
		else {
```

```
                str += "off";
            }

            return str;

        }
}

class tester{
        public static void main(String[]args) {
            Lamp flicker = new Switch();
        }
        }
```