Testing Lab 3

Abdelrahman Ibrahim

19P3049

Question 1:

Code:

```
public class coffeeMachine {
    public boolean isPowerOn = false;
    public boolean isLightOn = false;
    public boolean isBusy = false;
    public float money;
    private final float price = 10;
    public int coffeeAmount;

    public String addMoney(float money){
        if(isPowerOn) {
            this.money = money;
            return "Successfully added money";
        }
        else{
            return "Turn Machine On first!";
        }
    }
    public String openMachine(){
        if(!isPowerOn) {
            isPowerOn = true;
            isLightOn = true;
            return "Power On!" + "Lights On!";

        }
        else{
            return "Error!";
        }
    }
    public String closeMachine(){
        if(isPowerOn) {
            isLightOn = false;
            isPowerOn = false;
            return "Lights Off!" + "Power Off!";

        }
        else{
            return "Error!";
        }

    }
    public String makeCoffee(int coffeeAmount){
        if(isPowerOn) {
            int i= 0;
            if (money >= price) {
                while(money>= price && coffeeAmount > 0){
                    money -= price;
                    isBusy = true;
                    coffeeAmount --;
                    i++;
                }
            }
```

```
            else {
                return "Not enough balance";
            }
            isBusy = false;
            return i+ " Cup Of coffee Made!";
        }
        else{
            return "Turn Machine On first!";
        }
    }



}
```

Test code:

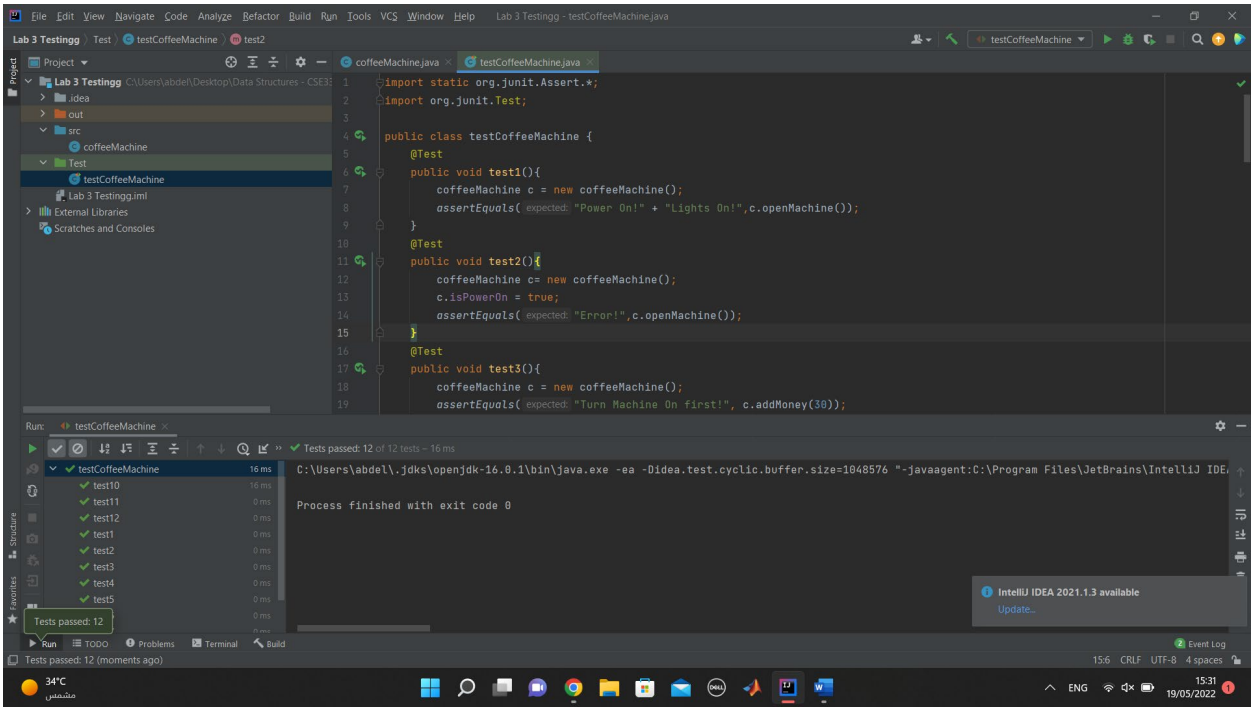
```
import static org.junit.Assert.*;
import org.junit.Test;

public class testCoffeeMachine {
    @Test
    public void test1(){
        coffeeMachine c = new coffeeMachine();
        assertEquals("Power On!" + "Lights On!",c.openMachine());
    }
    @Test
    public void test2(){
        coffeeMachine c= new coffeeMachine();
        c.isPowerOn = true;
        assertEquals("Error!",c.openMachine());
    }
    @Test
    public void test3(){
        coffeeMachine c = new coffeeMachine();
        assertEquals("Turn Machine On first!", c.addMoney(30));
    }
    @Test
    public void test4(){
        coffeeMachine c = new coffeeMachine();
        c.openMachine();
        assertEquals("Successfully added money",c.addMoney(30));
    }
    @Test
    public void test5(){
        coffeeMachine c = new coffeeMachine();
        c.openMachine();
        c.closeMachine();
        assertEquals("Turn Machine On first!",c.addMoney(30));
    }
    @Test
    public void test6(){
        coffeeMachine c = new coffeeMachine();
        assertEquals("Power On!" + "Lights On!",c.openMachine());
```

```
        assertEquals("Successfully added money",c.addMoney(30));
    }
    @Test
    public void test7(){
        coffeeMachine c = new coffeeMachine();
        assertEquals("Power On!" + "Lights On!",c.openMachine());
        assertEquals("Successfully added money",c.addMoney(30));
        assertEquals("Lights Off!" + "Power Off!",c.closeMachine());
    }
    @Test
    public void test8(){
        coffeeMachine c = new coffeeMachine();
        assertEquals("Turn Machine On first!",c.makeCoffee(2));
    }
    @Test
    public void test9(){
        coffeeMachine c = new coffeeMachine();
        assertEquals("Power On!" + "Lights On!",c.openMachine());
        assertEquals("Successfully added money",c.addMoney(30));
        assertEquals(3+ " Cup Of coffee Made!",c.makeCoffee(3));
    }
    @Test
    public void test10(){
        coffeeMachine c = new coffeeMachine();
        assertEquals("Power On!" + "Lights On!",c.openMachine());
        assertEquals("Successfully added money",c.addMoney(30));
        assertEquals(3+ " Cup Of coffee Made!",c.makeCoffee(4));
    }
    @Test
    public void test11(){
        coffeeMachine c = new coffeeMachine();
        assertEquals("Power On!" + "Lights On!",c.openMachine());
        assertEquals("Successfully added money",c.addMoney(30));
        assertEquals(1+ " Cup Of coffee Made!",c.makeCoffee(1));
    }
    @Test
    public void test12(){
        coffeeMachine c = new coffeeMachine();
        assertEquals("Power On!" + "Lights On!",c.openMachine());
        assertEquals("Successfully added money",c.addMoney(30));
        assertEquals(2+ " Cup Of coffee Made!",c.makeCoffee(2));
        assertEquals("Lights Off!" + "Power Off!",c.closeMachine());
    }

}
```

Output:


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Lab 3 Testingg - testCoffeeMachine.java
Lab 3 Testingg
Test
testCoffeeMachine
test2
coffeeMachine.java
testCoffeeMachine.java
Project
Lab 3 Testingg C:\Users\abdel\Desktop\Data Structures - CSE33
.idea
out
src
coffeeMachine
Test
testCoffeeMachine
Lab 3 Testingg.iml
External Libraries
Scratches and Consoles
import static org.junit.Assert.*;
import org.junit.Test;
public class testCoffeeMachine {
@Test
public void test1(){
coffeeMachine c = new coffeeMachine();
assertEquals(expected: "Power On!" + "Lights On!",c.openMachine());
}
@Test
public void test2(){
coffeeMachine c= new coffeeMachine();
c.isPowerOn = true;
assertEquals(expected: "Error!",c.openMachine());
}
@Test
public void test3(){
coffeeMachine c = new coffeeMachine();
assertEquals(expected: "Turn Machine On first!", c.addMoney(30));
Run: testCoffeeMachine
Tests passed: 12 of 12 tests – 16 ms
testCoffeeMachine 16 ms
test10 16 ms
test11 0 ms
test12 0 ms
test1 0 ms
test2 0 ms
test3 0 ms
test4 0 ms
test5 0 ms
C:\Users\abdel\.jdks\openjdk-16.0.1\bin\java.exe -ea -Didea.test.cyclic.buffer.size=1048576 "-javaagent:C:\Program Files\JetBrains\IntelliJ IDE
Process finished with exit code 0
IntelliJ IDEA 2021.1.3 available
Update...
Tests passed: 12
Run
TODO
Problems
Terminal
Build
Event Log
Tests passed: 12 (moments ago)
15:6 CRLF UTF-8 4 spaces
34°C
مشمس
ENG
15:31
19/05/2022

Question 2

Code:

```
public class ATM {

    public float accountBalance = 0;
    public boolean isAuth = false;
    public boolean isValidatedCard = true;
    public boolean isValidPassword = true;

    public String enterCard(){
        if(isValidatedCard){
            return "Validated card";
        }
        else{
            return "Card not validated";
        }
    }

    public String enterPassword(){
        if(isValidatedCard){
            System.out.println("Enter Password");
            if(isValidPassword){
                isAuth = true;
                return "Successful login";
            }
            else{
                return "Wrong password";
            }
        }
        else{
            return "Enter card first";
        }
    }

    public String deposit(float money){
        if(isAuth){
            if(money >= 50 && money <=50000){
                accountBalance += money;
                return "Successful deposit";
            }
            else{
                return "Enter valid amount";
            }
        }
        else{
            return "Not allowed";
        }
    }
    public String withdraw(float amount){
        if(isAuth){
            if(amount <= accountBalance){
                accountBalance -= amount;
                return "Successful withdraw";
```

```
            }
            else{
                return "Not enough balance";
            }
        }
        else{
            return "Not allowed";
        }
    }
    public String removeCard(){
        if(isValidatedCard){
            return "Card removed";
        }
        else{
            return "Not allowed";
        }
    }

}
```

Test Code:

```
import org.junit.Test;

import static org.junit.Assert.*;

public class testATM {


    @Test

    public void test1(){

        ATM a = new ATM();

        a.isValidatedCard = false;

        assertEquals("Card not validated",a.enterCard());

    }

    @Test

    public void test2(){

        ATM a = new ATM();
```

```
        assertEquals("Validated card",a.enterCard());
    }
    @Test
    public void  test3(){
        ATM a = new ATM();
        assertEquals("Successful login",a.enterPassword());
    }
    @Test
    public void  test4(){
        ATM a = new ATM();
        assertEquals("Validated card",a.enterCard());
        a.isValidPassword = false;
        assertEquals("Wrong password",a.enterPassword());
    }
    @Test
    public void  test5(){
        ATM a = new ATM();
        assertEquals("Validated card",a.enterCard());
        assertEquals("Successful login",a.enterPassword());
        assertEquals("Not enough balance",a.withdraw(10));
    }
    @Test
    public void  test6(){
        ATM a = new ATM();
        assertEquals("Validated card",a.enterCard());
        assertEquals("Successful login",a.enterPassword());
        assertEquals("Enter valid amount",a.deposit(20));
        assertEquals("Not enough balance",a.withdraw(10));
```

```
    }
    @Test
    public void  test7(){
        ATM a = new ATM();
        assertEquals("Validated card",a.enterCard());
        assertEquals("Successful login",a.enterPassword());
        assertEquals("Successful deposit",a.deposit(50));
        assertEquals("Not enough balance",a.withdraw(10));
    }
    @Test
    public void  test8(){
        ATM a = new ATM();
        assertEquals("Validated card",a.enterCard());
        assertEquals("Successful login",a.enterPassword());
        assertEquals("Successful deposit",a.deposit(1000));
        assertEquals("Successful withdraw",a.withdraw(10));
    }
    @Test
    public void  test9(){
        ATM a = new ATM();
        assertEquals("Validated card",a.enterCard());
        assertEquals("Successful login",a.enterPassword());
        assertEquals("Enter valid amount",a.deposit(50001));
        assertEquals("Not enough balance",a.withdraw(10));
    }
    @Test
    public void  test10(){
        ATM a = new ATM();
```

```
        assertEquals("Validated card",a.enterCard());
        assertEquals("Successful login",a.enterPassword());
        assertEquals("Successful deposit",a.deposit(1000));
        assertEquals("Successful withdraw",a.withdraw(500));
        assertEquals("Successful withdraw",a.withdraw(500));
    }
    @Test
    public void  test11(){
        ATM a = new ATM();
        assertEquals("Validated card",a.enterCard());
        assertEquals("Successful login",a.enterPassword());
        assertEquals("Successful deposit",a.deposit(1000));
        assertEquals("Successful withdraw",a.withdraw(500));
        assertEquals("Not enough balance",a.withdraw(501));
    }
    @Test
    public void  test12(){
        ATM a = new ATM();
        assertEquals("Validated card",a.enterCard());
        assertEquals("Successful login",a.enterPassword());
        assertEquals("Successful deposit",a.deposit(3000));
        assertEquals("Successful withdraw",a.withdraw(1000));
        assertEquals("Not enough balance",a.withdraw(4000));
        assertEquals("Successful withdraw",a.withdraw(2000));
    }



}
```

Output:

Questions 3:

```java
public class digitalWatch {
    public boolean isOpen = false;
    public boolean isTimerSet = false;
    public float timer = 0;

    public String turnOn(){
        if(!isOpen){
            isOpen = true;
            return "Turned On!";
        }
        else{
            return  "Already turned on";
        }
    }
    public String setTimer(float time){
        if(isOpen){
            timer = time;
            isTimerSet = true;
            return "Timer set";
        }
        else{
            return "Watch os closed";
        }
    }
    public String doSomething(){
        if(isTimerSet){
            return "Wait for timer to end";
        }
        else{
            return "Accepted";
        }
    }
    public String turnOff(){
        if(isOpen){
            isOpen = false;
            return "Turned off";
        }
        else{
            return "Error";
        }
```

```
    }
}
```

```
import org.junit.Test;
import static org.junit.Assert.*;
public class testDigitalWatch {
    @Test
    public void test1(){
        digitalWatch d = new digitalWatch();
        assertEquals("Turned On!",d.turnOn());
    }
    @Test
    public void test2(){
        digitalWatch d = new digitalWatch();
        assertEquals("Turned On!",d.turnOn());
        assertEquals("Already turned on",d.turnOn());
    }
    @Test
    public void test3(){
        digitalWatch d = new digitalWatch();
        assertEquals("Turned On!",d.turnOn());
        assertEquals("Already turned on",d.turnOn());
        assertEquals("Turned off",d.turnOff());
    }
    @Test
    public void test4(){
        digitalWatch d = new digitalWatch();
        assertEquals("Turned On!",d.turnOn());
        assertEquals("Timer set",d.setTimer(10));
    }
    @Test
    public void test5(){
        digitalWatch d = new digitalWatch();
        assertEquals("Turned On!",d.turnOn());
        assertEquals("Timer set",d.setTimer(10));
        assertEquals("Wait for timer to end",d.doSomething());
    }
    @Test
    public void test6(){
        digitalWatch d = new digitalWatch();
        assertEquals("Turned On!",d.turnOn());
        assertEquals("Timer set",d.setTimer(10));
        d.isTimerSet = false;
        assertEquals("Accepted",d.doSomething());
    }
    @Test
    public void test7(){
        digitalWatch d = new digitalWatch();
        assertEquals("Turned On!",d.turnOn());
        assertEquals("Timer set",d.setTimer(10));
        d.isTimerSet = false;
        assertEquals("Accepted",d.doSomething());
        assertEquals("Turned off",d.turnOff());
    }
    @Test
```

```
    public void test8(){
        digitalWatch d = new digitalWatch();
        assertEquals("Turned On!",d.turnOn());
        assertEquals("Timer set",d.setTimer(10));
        d.isTimerSet = false;
        assertEquals("Accepted",d.doSomething());
        d.isOpen = false;
        assertEquals("Error",d.turnOff());
    }



}
```

Output: