LAB ASSIGNMENT 3

Karen Alber Farid
19P8948

May 21, 2022

—

Software Testing

—

Eng. Adham

ATM Machine

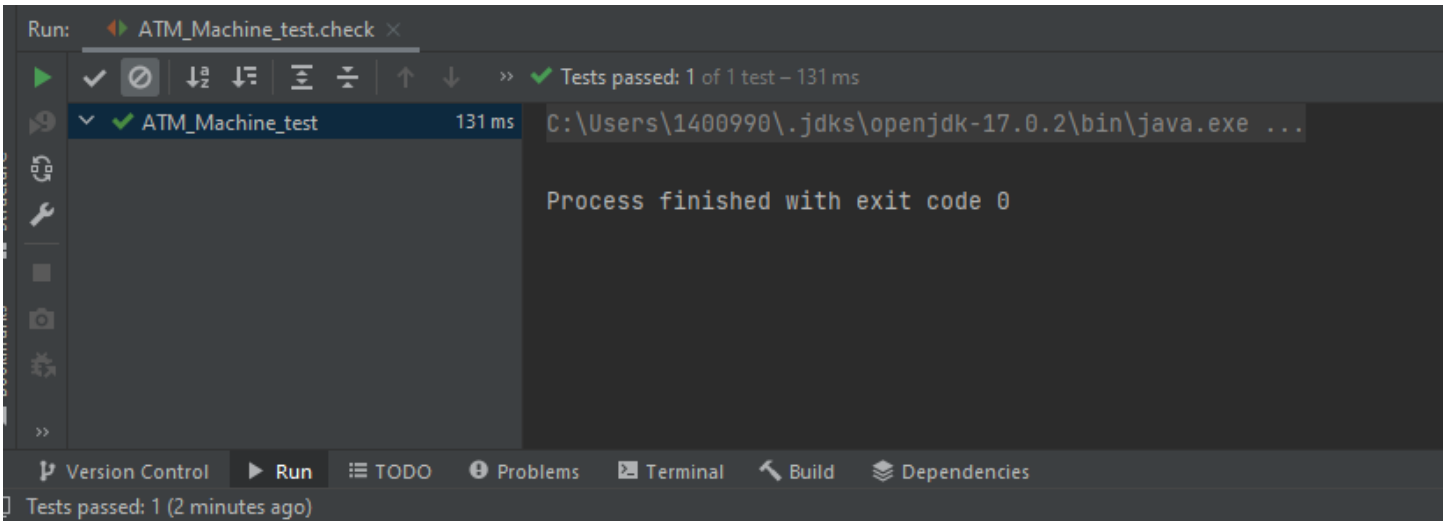
Code:

```
public class ATM_Machine {
   public double ATM(int choice, double withdraw, double deposit, double balance) {
      switch (choice) {
         case 1:

            if (balance >= withdraw) {
               balance = balance - withdraw;
            } else {
               System.out.println("Insufficient Balance");
            }

            break;

         case 2:

            balance = balance + deposit;
            break;

         case 3:
            break;

         case 4:
            //exit from the menu
            System.exit(0);
      }
      return balance;
   }
}
```

Test:

```
import org.junit.Test;

import static org.junit.Assert.*;
import static org.junit.jupiter.api.Assertions.assertEquals;


public class ATM_Machine_test {



    @Test
    public void check(){

        ATM_Machine atm1 = new ATM_Machine();  //after deposit
        ATM_Machine atm2 = new ATM_Machine(); //after withdraw
        ATM_Machine atm4 = new ATM_Machine(); //view balance

        assertEquals(1100 , atm1.ATM(2,0,100,1000));
        assertEquals(900 , atm2.ATM(1,100,0,1000));
        assertEquals(1000 , atm4.ATM(3,0,0,1000));
    }

}
```

Output:



Coffee Machine

Code:

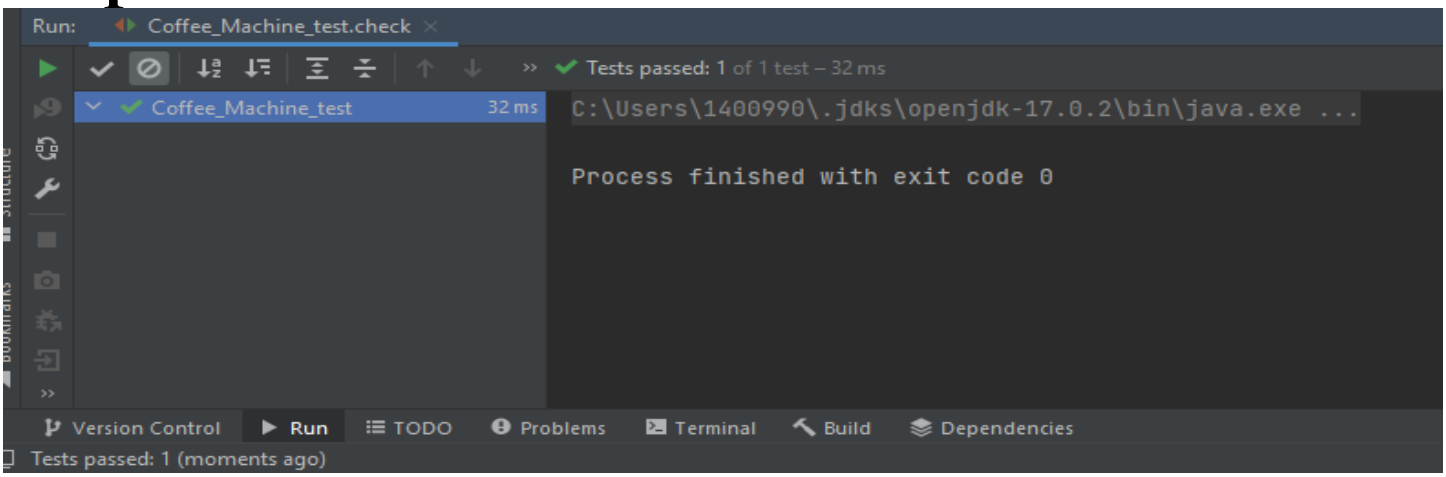
```
public class Coffee_Machine {

    public double coffee(int t, int sugar) {
        int coffee_powder, milk, water;
        double price = 0;
        int Coffee_Count = 0;

        switch (t) {
            // Black coffee
            case 1:
                coffee_powder = 1;
                milk = 0;
                water = 1;
                price = 20;
                break;

            // Milk coffee
            case 2:
                coffee_powder = 1;
                milk = 1;
                water = 0;
                price = 40;
                break;

            // Milk and Water coffee
            case 3:
                coffee_powder = 1;
                milk = 1;
                water = 1;
                price = 30;
                break;

            // Bottle of water
            case 0:
                coffee_powder = 0;
                milk = 0;
                water = 1;
```

```
            price = 10;
        }
        return price;
    }
}
```

Test:

```
import org.junit.Test;
import static org.junit.Assert.*;
import static org.junit.jupiter.api.Assertions.assertEquals;

public class Coffee_Machine_test {
    @Test
    public void check() {

        Coffee_Machine cm1 = new Coffee_Machine();  //Black Coffee
        Coffee_Machine cm2 = new Coffee_Machine(); //Milk Coffee
        Coffee_Machine cm3 = new Coffee_Machine(); //Milk and Water Coffee
        Coffee_Machine cm4 = new Coffee_Machine(); //Bottle of water

        assertEquals(20, cm1.coffee(1, 3));
        assertEquals(40, cm2.coffee(2, 1));
        assertEquals(30, cm3.coffee(3, 0));
        assertEquals(10, cm3.coffee(0, 4));

    }
}
```

Output:



Digital Watch

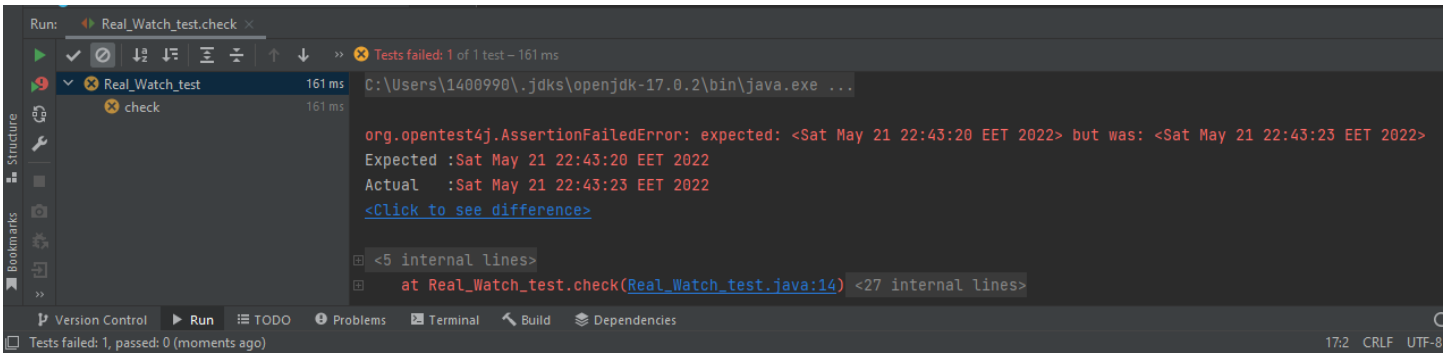
Code:

```
import java.util.*;
import java.text.*;


public class Real_Watch {
   public Date RealWatch() {

      String calen;

      int hours = 0, minutes = 0, seconds = 0;
      String timeString = "";

      Calendar cal = Calendar.getInstance();
      hours =cal.get(Calendar.HOUR_OF_DAY );
      if(hours >12)
         hours -=12;
      minutes =cal.get(Calendar.MINUTE );
      seconds =cal.get(Calendar.SECOND );

      SimpleDateFormat formatter = new SimpleDateFormat("hh:mm:ss");
      Date date = cal.getTime();
      timeString =formatter.format(date );

      calen = formatter + "\n" + timeString;

      return date;
   }
}
```

Test:

```
import org.junit.Assert;
import org.junit.Test;
import static org.junit.Assert.*;
import static org.junit.jupiter.api.Assertions.assertEquals;

public class Real_Watch_test {

    @Test
    public void check() {

        Real_Watch rw1 = new Real_Watch();

        // the expected of assertEquals is to enter current date and time as follows
        assertEquals("Sat May 21 07:19:00", rw1.RealWatch());

    }
}
```

Output:



The only issue here is that the expected had a 3 seconds delay than the actual. No issues other than that and the code with its test case is working excellently except for these 3 seconds delay.