Name: Shady Osama , ID: 19p2602
Section :2 ,Group :1

Question 1:

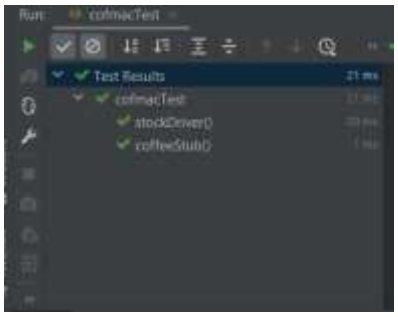
```
public class cofmac
{
     money f = new money();
    stock s = new stock();
    public void coffee(int s){
        this.s.addToStock(s);
    }
    void despence(int x){
        while(x>0 && s.isEmpty()){
            System.out.println("coffee");
            f.addOnePound();
            s.takeFromStock();
            x--;
        }
        if (x!=0)
            System.out.println("machine empty");
    }
    void status(){
        System.out.println("there are "+ f.isEmpty()+ " coins \n there are"
                +s.isEmpty()+" coffees in stock\n");
    }
    public static void main(String args[]){
    }
}
```

```
public class stock {
    int inventory=10;
    boolean isEmpty(){
        if(inventory>0)
            return true;
        else
            return false;
    }
    void addToStock(int add){
        inventory +=add;
    }
    void takeFromStock(){
        inventory -=1;
    }
    void redeem(){
        inventory
        =0;
    }
    int showAvailableStock(){
        return inventory;
    }
```

```
public class money {
    int c = 0;
    boolean isEmpty(){
        if(c>0)
            return true;
        else
            return false;
    }
    void redeem(int c){
        c =0;
    }
    void addOnePound(){
        c++;
    }
}
```

```

import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

class cofmacTest {
        cofmac x;
        stock k;
        money m;
        String s;
        @BeforeEach
        public void init() {
            x = new cofmac();
            k = new stock();
            m = new money();
        }
        class coinInsert{
            int insert(int x){
                if(x>0)
                    return x;
                else
                    return 0;
            }
        }
        @Test
        public void coffeeStub(){
            coinInsert y = new coinInsert();
            x.despence(y.insert(5));
            assertEquals(x.s.money,5);
        }
        class doubleCoffeeCheck{
            String check(int x){
                if(x%2==0)
                    return "double coffee can be made ";
                else
                    return " double coffee cannot be made ";
            }
        }
        @Test
```

```java
        public void stock_Driver(){
            doubleCoffee y = new doubleCoffee();
            s = y.check(k.showAvailableStock());
            assertEquals(s,"double coffee can be made ");
            k.takeFromStock();
            s = y.check(k.showAvailableStock());
            assertEquals(s,"double coffee cannot be made ");

        }
        @AfterEach
        public void clean(){
            x=null ;
        }
}
```

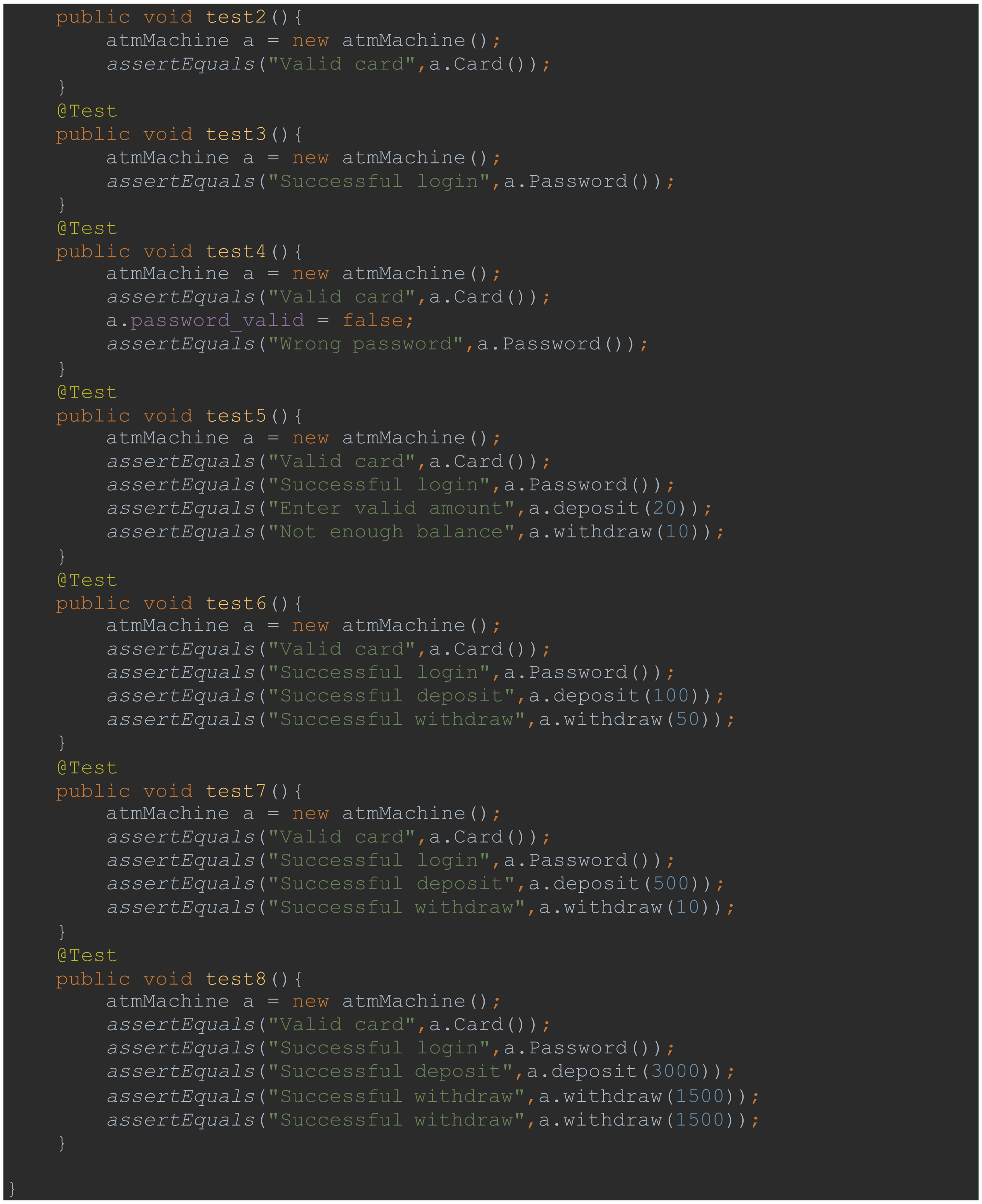


```
Run:   cofmacTest
Test Results          21 ms
  cofmacTest          21 ms
    stockDriver()     20 ms
    coffeeStub()       1 ms
```

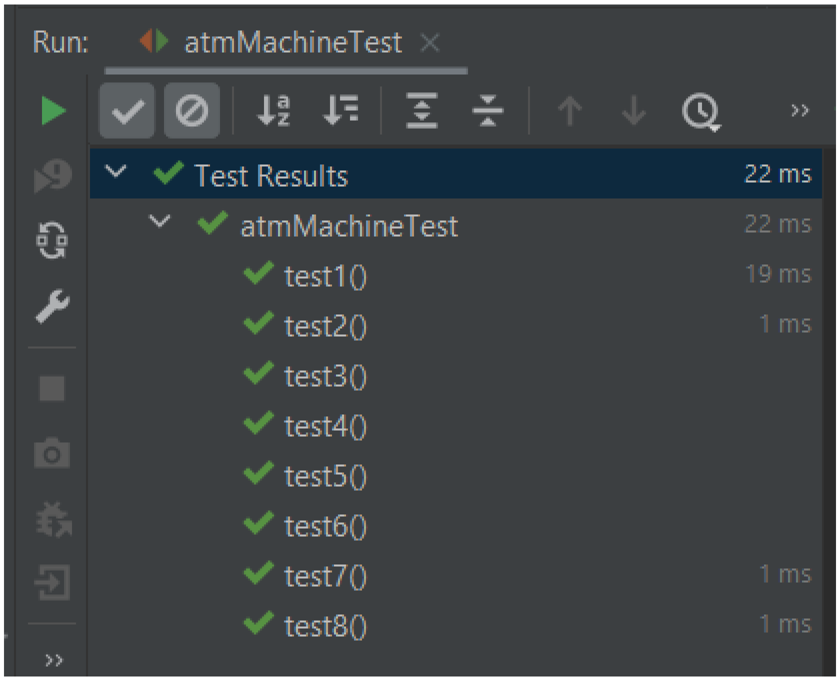
Question 2:

```java
public class atmMachine {
    public float Balance = 0;
    public boolean authentic = false;
    public boolean valid_card = true;
    public boolean password_valid = true;
    public String Card(){
        if(valid_card){
            return "Valid card";
        }
        else{
            return "Card not valid";
        }
    }
    public String Password(){
        if(valid_card){
            System.out.println("Enter Password");
            if(password_valid){
                authentic = true;
                return "Successful login";
            }
            else{
                return "Wrong password";
            }
        }
        else{
            return "Enter card first";
        }
    }
    public String deposit(float money){
        if(authentic){
            if(money >= 50 && money <=50000){
                Balance += money;
                return "Successful deposit";
            }
            else{
                return "Enter valid amount";
            }
        }
        else{
            return "Not allowed";
        }
    }
    public String withdraw(float amount){
        if(authentic){
            if(amount <= Balance){
                Balance -= amount;
                return "Successful withdraw";
            }
            else{
                return "Not enough balance";
            }
        }
        else{
            return "Not allowed";
        }
    }
    public String removeCard(){
        if(valid_card){
            return "Card removed";
        }
        else{
            return "Not allowed";
        }
    }
}
```

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

class atmMachineTest {
    @Test
    public void test1(){
        atmMachine a = new atmMachine();
        a.valid_card =  false;
        assertEquals("Card not valid",a.Card());
    }
    @Test
```



```java
    public void test2(){
        atmMachine a = new atmMachine();
        assertEquals("Valid card",a.Card());
    }
    @Test
    public void test3(){
        atmMachine a = new atmMachine();
        assertEquals("Successful login",a.Password());
    }
    @Test
    public void test4(){
        atmMachine a = new atmMachine();
        assertEquals("Valid card",a.Card());
        a.password_valid = false;
        assertEquals("Wrong password",a.Password());
    }
    @Test
    public void test5(){
        atmMachine a = new atmMachine();
        assertEquals("Valid card",a.Card());
        assertEquals("Successful login",a.Password());
        assertEquals("Enter valid amount",a.deposit(20));
        assertEquals("Not enough balance",a.withdraw(10));
    }
    @Test
    public void test6(){
        atmMachine a = new atmMachine();
        assertEquals("Valid card",a.Card());
        assertEquals("Successful login",a.Password());
        assertEquals("Successful deposit",a.deposit(100));
        assertEquals("Successful withdraw",a.withdraw(50));
    }
    @Test
    public void test7(){
        atmMachine a = new atmMachine();
        assertEquals("Valid card",a.Card());
        assertEquals("Successful login",a.Password());
        assertEquals("Successful deposit",a.deposit(500));
        assertEquals("Successful withdraw",a.withdraw(10));
    }
    @Test
    public void test8(){
        atmMachine a = new atmMachine();
        assertEquals("Valid card",a.Card());
        assertEquals("Successful login",a.Password());
        assertEquals("Successful deposit",a.deposit(3000));
        assertEquals("Successful withdraw",a.withdraw(1500));
        assertEquals("Successful withdraw",a.withdraw(1500));
    }

}
```



```
Test Results          22 ms
  atmMachineTest      22 ms
    test1()           19 ms
    test2()            1 ms
    test3()
    test4()
    test5()
    test6()
    test7()            1 ms
    test8()            1 ms
```

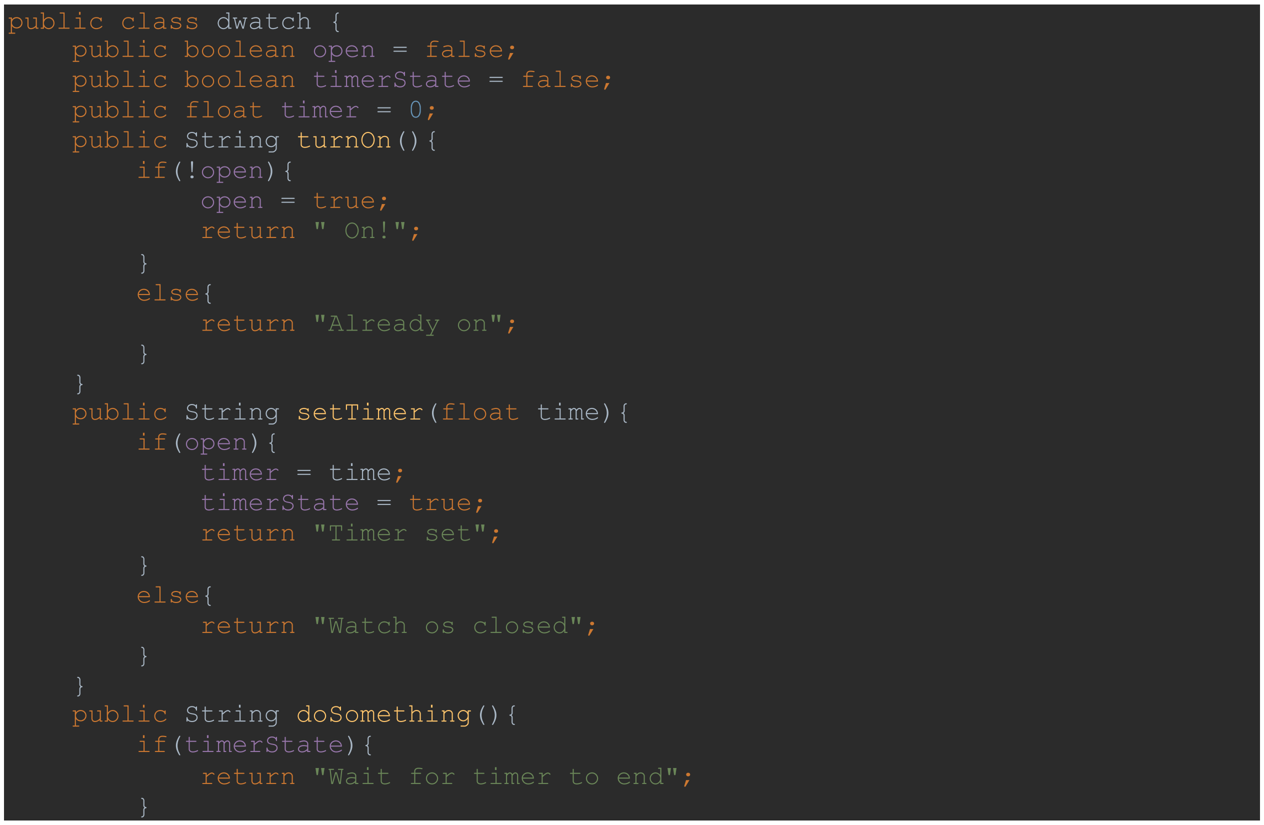
Another Atm Machine :

```
import java.util.Scanner;

public class AtmPin {

  public static boolean validPIN(int user, int orignal){


     return user==orignal;

  }

  public static int getPin(Scanner sc){


     System.out.print("Enter PIN: ");

     int pin = sc.nextInt();

     return pin;

  }

}
```

Test :

```
package atm;

import java.util.Scanner;

public class ATMTester {
    public static void main(String[] args) {



         Scanner keyboard = new Scanner(System.in);

         int i = 0, userpin;

        int PIN = 1234;
        while(i< 3){



            userpin = AtmPin.getPin(keyboard);



            if(AtmPin.validPIN(userpin, PIN)){

               System.out.println("Your PIN is correct");

               System.exit(0);

            }
            else {
               System.out.println("Your PIN is incorrect");
            }


            i++;

         }



         System.out.println("Your Bank Card is blocked");

      }

}
```

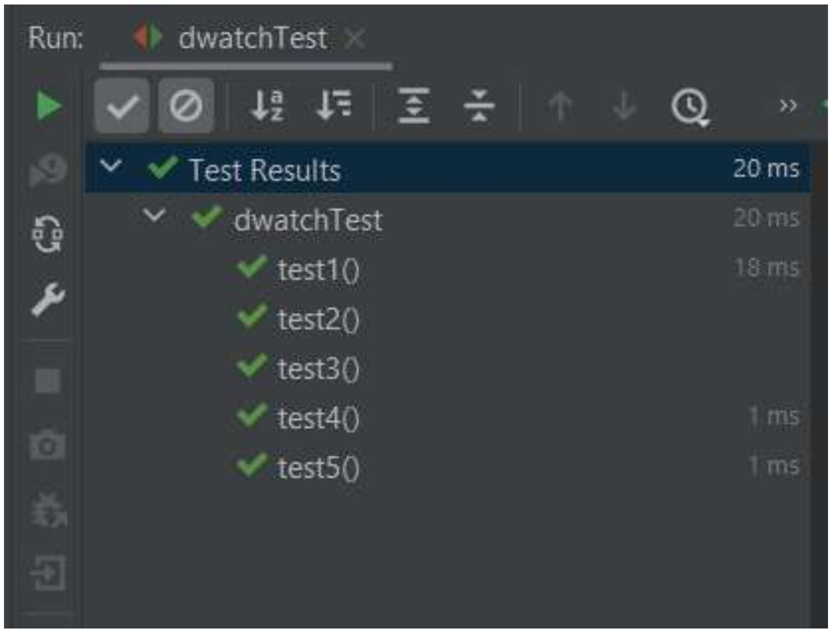
Question 3:



```java
public class dwatch {
    public boolean open = false;
    public boolean timerState = false;
    public float timer = 0;
    public String turnOn(){
        if(!open){
            open = true;
            return " On!";
        }
        else{
            return "Already on";
        }
    }
    public String setTimer(float time){
        if(open){
            timer = time;
            timerState = true;
            return "Timer set";
        }
        else{
            return "Watch os closed";
        }
    }
    public String doSomething(){
        if(timerState){
            return "Wait for timer to end";
        }
        else{
            return "Accepted";
        }
    }
    public String turnOff(){
        if(open){
            open = false;
            return "Turned off";
        }
        else{
            return "Error";
        }
    }}
```

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

class dwatchTest {
    @Test
    void test1() {
        dwatch d = new dwatch();
        assertEquals(" On!",d.turnOn());
    }
    @Test
    public void test2(){
        dwatch d = new dwatch();
        assertEquals(" On!",d.turnOn());
        assertEquals("Already on",d.turnOn());
    }
    @Test
    public void test3(){
        dwatch d = new dwatch();
        assertEquals(" On!",d.turnOn());
        assertEquals("Already on",d.turnOn());
        assertEquals("Turned off",d.turnOff());
    }

    @Test
    public void test4(){
        dwatch d = new dwatch();
        assertEquals(" On!",d.turnOn());
        assertEquals("Timer set",d.setTimer(10));
        assertEquals("Wait for timer to end",d.doSomething());
    }

    @Test
    public void test5(){
        dwatch d = new dwatch();
        assertEquals(" On!",d.turnOn());
        assertEquals("Timer set",d.setTimer(10));
        d.timerState = false;
        assertEquals("Accepted",d.doSomething());
        assertEquals("Turned off",d.turnOff());
    }
}
```

```
import javax.swing.*;
import java.awt.*;
import java.text.*;
import java.util.*;
public class DigitalWatch implements Runnable{
    JFrame f;
    Thread t=null;
    int hours=0, minutes=0, seconds=0;
    String timeString = "";
    JButton b;

    DigitalWatch(){
        f=new JFrame();

        t = new Thread(this);
        t.start();

        b=new JButton();
        b.setBounds(100,100,100,50);

        f.add(b);
        f.setSize(300,400);
        f.setLayout(null);
        f.setVisible(true);
    }

    public void run() {
        try {
            while (true) {

                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 );

                printTime();

                t.sleep( 1000 );  // interval given in milliseconds
            }
        }
        catch (Exception e) { }
    }

public String  runStub (){

String hours ="1" ;
String minutes = "12" ;
String seconds ="34";
int hour = Integer.parseInt(hours);
    if ( hour > 12 ) {
hour -=12 ;
    }
hours = String.valueOf(hour);
        timeString = hours+":"+minutes+":"+seconds;
    printTime();
        return timeString;

    }


    public void printTime(){
        b.setText(timeString);
    }

    public static void main(String[] args) {
        new DigitalWatch();


    }
}
```



```
Run:   dwatchTest
✓ Test Results      20 ms
  ✓ dwatchTest      20 ms
    ✓ test1()       18 ms
    ✓ test2()
    ✓ test3()
    ✓ test4()        1 ms
    ✓ test5()        1 ms
```

DW :

```
import javax.swing.*;
import java.awt.*;
import java.text.*;
import java.util.*;
public class DigitalWatch implements Runnable{
    JFrame f;
    Thread t=null;
    int hours=0, minutes=0, seconds=0;
    String timeString = "";
    JButton b;

    DigitalWatch(){
        f=new JFrame();

        t = new Thread(this);
        t.start();

        b=new JButton();
        b.setBounds(100,100,100,50);

        f.add(b);
        f.setSize(300,400);
        f.setLayout(null);
        f.setVisible(true);
    }

    public void run() {
        try {
            while (true) {

                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 );

                printTime();

                t.sleep( 1000 );  // interval given in milliseconds
            }
        }
        catch (Exception e) { }
    }

public String  runStub (){

String hours ="1" ;
String minutes = "12" ;
String seconds ="34";
int hour = Integer.parseInt(hours);
    if ( hour > 12 ) {
hour -=12 ;
    }
hours = String.valueOf(hour);
        timeString = hours+":"+minutes+":"+seconds;
    printTime();
        return timeString;
    }


    public void printTime(){
        b.setText(timeString);
    }

    public static void main(String[] args) {
        new DigitalWatch();


    }
}
```