EXP1----------ATM System

```
package EXP1;

import java.util.*;
import java.lang.*;

public class ATMmain {
    private static int pin = 7563, lim = 0, upin, accountType, balance = 5000, operation, wAmt, dAmt;

    public static void main(String args[]) {
        Scanner in = new Scanner(System.in);
        System.out.println("Welcome to the ICICI Bank ATM!");
        System.out.println("Please insert your ATM card.");
        while (lim <= 3) {
            System.out.println("Please enter your PIN.");
            upin = in.nextInt();
            if (upin == pin) {
                while (true) {
                    System.out.println("Select your account type.\n1. Savings\n2. Current");
                    accountType = in.nextInt();
                    if (accountType == 1) {
                        System.out.println("You have selected Savings Account.");
                        break;
                    } else if (accountType == 2) {
                        System.out.println("You have selected Current Account.");
                        break;
                    } else {
                        System.out.println("You have entered an invalid input. Try again.");
                    }
                }
                while (true) {
                    System.out.println("Select an operation:\n1. Check Balance\n2. Withdrawal\n3. Deposit\n4. Exit");
                    operation = in.nextInt();
                    if (operation == 1) {
                        System.out.println("Your current balance is: " + balance);

                    } else if (operation == 2) {
                        System.out.println("Enter amount to be withdrawn: ");
                        wAmt = in.nextInt();
                        if (wAmt > balance) {
                            System.out.println("The amount to be withdrawn is greater than current balance.");
                        } else if (wAmt > 0) {
                            balance -= wAmt;
                            System.out.println("Please collect your cash. Thank you.");
                        } else {
                            System.out.println("Please enter an amount greater than zero.");
                        }
                    } else if (operation == 3) {
                        System.out.println("Enter amount to be deposited: ");
                        dAmt = in.nextInt();
                        if (dAmt > 0) {
                            balance += dAmt;
                            System.out.println("Thank you.");
                        } else {
                            System.out.println("Please enter an amount greater than zero.");
                        }
                    } else if (operation == 4) {
                        System.out.println("Thank you for banking with ICICI Bank. Have a nice day!");
                        System.exit(0);
```

```
□□□□□□break;
□□□□□} else {
□□□□□□System.out.println("You have entered an invalid input. Try again.");
□□□□□}
□□□□}
□□□} else {
□□□□lim++;
□□□□if (lim == 1) {
□□□□□System.out.println("Incorrect PIN. You have 2 more chances after which your card will be blocked.");
□□□□}
□□□□if (lim == 2) {
□□□□□System.out.println("Incorrect PIN. You have 1 more chance after which your card will be blocked.");
□□□□}
□□□□if (lim == 3) {

□□□□□System.out.println("Incorrect PIN. Your card has been blocked.");
□□□□□System.out.println("Please take out your card. Thank you.");
□□□□□System.exit(0);
□□□□}
□□□}
□□}
□}
}
```

EXP 2 ----------The Triangle Problem

```
package EXP2;

import java.util.*;

public class triangle {
□public static void main(String[] args) {
□□Scanner in = new Scanner(System.in);
□□int a, b, c;
□□while (true) {
□□□System.out.print("Enter value of 1st side: ");
□□□a = in.nextInt();
□□□System.out.print("Enter value of 2nd side: ");
□□□b = in.nextInt();
□□□System.out.print("Enter value of 3rd side: ");
□□□c = in.nextInt();
□□□if (a >= 1 && a <= 200 && b >= 1 && b <= 200 && c >= 1 && c <= 200) {
□□□□if ((a < b + c) && (b < a + c) && (c < b + a)) {
□□□□□if (a == b && b == c)
□□□□□□System.out.println("Given dimensions form an equilateral triangle!");
□□□□□else if (a == b || b == c || c == a)
□□□□□□System.out.println("Given dimensions form an isosceles triangle!");
□□□□□else
□□□□□□System.out.println("Given dimensions form a scalene triangle!");
□□□□□break;
□□□□} else {
□□□□□System.out.println("Given dimensions do not form a triangle!");
□□□□□break;
□□□□}
□□□}

□□□else {
```

```
    System.out.println("Enter a valid input!\n");
  }
 }
}
```

EXP3----------Boundary Value Analysis (BVA) for the NextDate Function

```
package EXP3;

import java.util.*;

public class date {
 public String nextDate(int d, int m, int y) {
  int nd, nm, ny;
  if (d > 31 || d < 1 || m > 12 || m < 1 || y < 1821 || y > 2021) {
   return ("Invalid date!");
  } else if (m == 2 || m == 4 || m == 6 || m == 9 || m == 11) {
   if (d == 31)
    return ("Invalid date!");
   else if (m == 2) {
    if (checkLeapYear(y)) {
     if (d > 29) {
      return ("Invalid date!");
     }
     if (d == 29) {
      nd = 1;
      nm = 3;
     } else {
      nd = ++d;
      nm = 2;
     }
    } else {
     if (d > 28) {
      return ("Invalid date!");
     }
     if (d == 28) {
      nd = 1;
      nm = 3;
     }

     else {
      nd = ++d;
      nm = 2;
     }
    }
    ny = y;
   } else {
    if (d == 30) {
     nd = 1;
     nm = ++m;
    } else {
     nd = ++d;
     nm = m;
    }
    ny = y;
   }
  } else {
```

```
   if (d == 31 && m != 12) {
    nd = 1;
    nm = ++m;
    ny = y;
   } else if (d == 31 && m == 12) {
    nd = 1;
    nm = 1;
    ny = ++y;
   } else {
    nd = ++d;
    nm = m;
    ny = y;
   }
  }
  return ("The next date is: " + nd + "-" + nm + "-" + ny);
 }

 public static boolean checkLeapYear(int year) {
  if (year % 400 == 0)
   return true;
  else if (year % 100 == 0)
   return false;
  else if (year % 4 == 0)
   return true;
  else
   return false;
 }
}


package EXP3;

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

import org.junit.jupiter.api.Test;

class NormalBVT {
 @Test
 public void test1() {
  date d1 = new date();
  assertEquals(d1.nextDate(15, 6, 1821), "The next date is: 16-6-1821");
 }

 @Test
 public void test2() {
  date d1 = new date();
  assertEquals(d1.nextDate(15, 6, 1822), "The next date is: 16-6-1822");
 }

 @Test
 public void test3() {
  date d1 = new date();
  assertEquals(d1.nextDate(15, 6, 1921), "The next date is: 16-6-1921");
 }

 @Test
 public void test4() {
  date d1 = new date();
  assertEquals(d1.nextDate(15, 6, 2020), "The next date is: 16-6-2020");
 }
```

```
    @Test
    public void test5() {
        date d1 = new date();
        assertEquals(d1.nextDate(15, 6, 2021), "The next date is: 16-6-2021");
    }

    @Test
    public void test6() {
        date d1 = new date();
        assertEquals(d1.nextDate(15, 1, 1921), "The next date is: 16-1-1921");
    }

    @Test
    public void test7() {
        date d1 = new date();
        assertEquals(d1.nextDate(15, 2, 1921), "The next date is: 16-2-1921");
    }

    @Test
    public void test8() {
        date d1 = new date();
        assertEquals(d1.nextDate(15, 11, 1921), "The next date is: 16-11-1921");
    }

    @Test
    public void test9() {
        date d1 = new date();
        assertEquals(d1.nextDate(15, 12, 1921), "The next date is: 16-12-1921");
    }

    @Test
    public void test10() {
        date d1 = new date();
        assertEquals(d1.nextDate(1, 6, 1921), "The next date is: 2-6-1921");
    }

    @Test
    public void test11() {
        date d1 = new date();
        assertEquals(d1.nextDate(2, 6, 1921), "The next date is: 3-6-1921");
    }

    @Test
    public void test12() {
        date d1 = new date();
        assertEquals(d1.nextDate(29, 6, 1921), "The next date is: 30-6-1921");
    }

    @Test
    public void test13() {
        date d1 = new date();
        assertEquals(d1.nextDate(30, 6, 1921), "The next date is: 1-7-1921");
    }

}


package EXP3;
```

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

import org.junit.jupiter.api.Test;

class Robust_WC_BVT {

	@Test
	public void test1() {
		date d1 = new date();
		assertEquals(d1.nextDate(0, 1, 1821), "Invalid date!");
	}

	@Test
	public void test2() {
		date d1 = new date();
		assertEquals(d1.nextDate(0, 1, 1822), "Invalid date!");
	}

	@Test
	public void test3() {
		date d1 = new date();
		assertEquals(d1.nextDate(0, 1, 1921), "Invalid date!");
	}

	@Test
	public void test4() {
		date d1 = new date();
		assertEquals(d1.nextDate(0, 1, 2020), "Invalid date!");
	}

	@Test
	public void test5() {
		date d1 = new date();
		assertEquals(d1.nextDate(0, 1, 2021), "Invalid date!");
	}

	@Test
	public void test6() {
		date d1 = new date();
		assertEquals(d1.nextDate(1, 1, 1821), "The next date is: 2-1-1821");
	}

	@Test
	public void test7() {
		date d1 = new date();
		assertEquals(d1.nextDate(1, 1, 1822), "The next date is: 2-1-1822");
	}

	@Test
	public void test8() {
		date d1 = new date();
		assertEquals(d1.nextDate(1, 1, 1921), "The next date is: 2-1-1921");
	}

	@Test
	public void test9() {
		date d1 = new date();
		assertEquals(d1.nextDate(1, 1, 2020), "The next date is: 2-1-2020");
	}
```

```
@Test
public void test10() {
    date d1 = new date();
    assertEquals(d1.nextDate(1, 1, 2021), "The next date is: 2-1-2021");
}

@Test
public void test11() {
    date d1 = new date();
    assertEquals(d1.nextDate(2, 1, 1821), "The next date is: 3-1-1821");
}

@Test
public void test12() {
    date d1 = new date();
    assertEquals(d1.nextDate(2, 1, 1822), "The next date is: 3-1-1822");
}

@Test
public void test13() {
    date d1 = new date();
    assertEquals(d1.nextDate(2, 1, 1921), "The next date is: 3-1-1921");
}

@Test
public void test14() {
    date d1 = new date();
    assertEquals(d1.nextDate(2, 1, 2020), "The next date is: 3-1-2020");
}

@Test
public void test15() {
    date d1 = new date();
    assertEquals(d1.nextDate(2, 1, 2021), "The next date is: 3-1-2021");
}

@Test
public void test16() {
    date d1 = new date();
    assertEquals(d1.nextDate(6, 1, 1821), "The next date is: 7-1-1821");
}

@Test
public void test17() {
    date d1 = new date();
    assertEquals(d1.nextDate(6, 1, 1822), "The next date is: 7-1-1822");
}

@Test
public void test18() {
    date d1 = new date();
    assertEquals(d1.nextDate(6, 1, 1921), "The next date is: 7-1-1921");
}

@Test
public void test19() {
    date d1 = new date();
    assertEquals(d1.nextDate(6, 1, 2020), "The next date is: 7-1-2020");
}
```

```
@Test
public void test20() {
    date d1 = new date();

    assertEquals(d1.nextDate(6, 1, 2021), "The next date is: 7-1-2021");
}

@Test
public void test21() {
    date d1 = new date();
    assertEquals(d1.nextDate(30, 1, 1821), "The next date is: 31-1-1821");
}

@Test
public void test22() {
    date d1 = new date();
    assertEquals(d1.nextDate(30, 1, 1822), "The next date is: 31-1-1822");
}

@Test
public void test23() {
    date d1 = new date();
    assertEquals(d1.nextDate(30, 1, 1921), "The next date is: 31-1-1921");
}

@Test
public void test24() {
    date d1 = new date();
    assertEquals(d1.nextDate(30, 1, 2020), "The next date is: 31-1-2020");
}

@Test
public void test25() {
    date d1 = new date();
    assertEquals(d1.nextDate(30, 1, 2021), "The next date is: 31-1-2021");
}

@Test
public void test26() {
    date d1 = new date();
    assertEquals(d1.nextDate(31, 1, 1821), "The next date is: 1-2-1821");
}

@Test
public void test27() {
    date d1 = new date();
    assertEquals(d1.nextDate(31, 1, 1822), "The next date is: 1-2-1822");
}

@Test
public void test28() {
    date d1 = new date();
    assertEquals(d1.nextDate(31, 1, 1921), "The next date is: 1-2-1921");
}

@Test
public void test29() {
    date d1 = new date();
    assertEquals(d1.nextDate(31, 1, 2020), "The next date is: 1-2-2020");
}
```

```
    @Test
    public void test30() {
        date d1 = new date();
        assertEquals(d1.nextDate(31, 1, 2021), "The next date is: 1-2-2021");
    }

    @Test
    public void test31() {
        date d1 = new date();
        assertEquals(d1.nextDate(32, 1, 1821), "Invalid date!");
    }

    @Test
    public void test32() {
        date d1 = new date();
        assertEquals(d1.nextDate(32, 1, 1822), "Invalid date!");
    }

    @Test
    public void test33() {
        date d1 = new date();
        assertEquals(d1.nextDate(32, 1, 1921), "Invalid date!");
    }

    @Test
    public void test34() {
        date d1 = new date();
        assertEquals(d1.nextDate(32, 1, 2020), "Invalid date!");
    }

    @Test
    public void test35() {
        date d1 = new date();
        assertEquals(d1.nextDate(32, 1, 2021), "Invalid date!");
    }
}
```

```
package EXP3;

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

import org.junit.jupiter.api.Test;

class RobustBVT {

    @Test
    public void test1() {
        date d1 = new date();
        assertEquals(d1.nextDate(15, 6, 1820), "Invalid date!");
    }

    @Test
    public void test2() {
        date d1 = new date();
        assertEquals(d1.nextDate(15, 6, 1821), "The next date is: 16-6-1821");
    }

    @Test
```

```
public void test3() {
    date d1 = new date();
    assertEquals(d1.nextDate(15, 6, 1822), "The next date is: 16-6-1822");
}

@Test
public void test4() {
    date d1 = new date();
    assertEquals(d1.nextDate(15, 6, 1921), "The next date is: 16-6-1921");
}

@Test
public void test5() {
    date d1 = new date();
    assertEquals(d1.nextDate(15, 6, 2020), "The next date is: 16-6-2020");
}

@Test
public void test6() {
    date d1 = new date();
    assertEquals(d1.nextDate(15, 6, 2021), "The next date is: 16-6-2021");
}

@Test
public void test7() {
    date d1 = new date();
    assertEquals(d1.nextDate(15, 6, 2022), "Invalid date!");
}

@Test
public void test8() {
    date d1 = new date();
    assertEquals(d1.nextDate(15, 0, 1921), "Invalid date!");
}

@Test
public void test9() {
    date d1 = new date();
    assertEquals(d1.nextDate(15, 1, 1921), "The next date is: 16-1-1921");
}

@Test
public void test10() {
    date d1 = new date();
    assertEquals(d1.nextDate(15, 2, 1921), "The next date is: 16-2-1921");
}

@Test
public void test11() {
    date d1 = new date();
    assertEquals(d1.nextDate(15, 11, 1921), "The next date is: 16-11-1921");
}

@Test
public void test12() {
    date d1 = new date();
    assertEquals(d1.nextDate(15, 12, 1921), "The next date is: 16-12-1921");
}

@Test
```

```
	public void test13() {
		date d1 = new date();
		assertEquals(d1.nextDate(15, 13, 1921), "Invalid date!");
	}

	@Test
	public void test14() {
		date d1 = new date();
		assertEquals(d1.nextDate(0, 6, 1921), "Invalid date!");
	}

	@Test
	public void test15() {
		date d1 = new date();
		assertEquals(d1.nextDate(1, 6, 1921), "The next date is: 2-6-1921");
	}

	@Test
	public void test16() {
		date d1 = new date();
		assertEquals(d1.nextDate(2, 6, 1921), "The next date is: 3-6-1921");
	}

	@Test
	public void test17() {
		date d1 = new date();
		assertEquals(d1.nextDate(29, 6, 1921), "The next date is: 30-6-1921");
	}

	@Test
	public void test18() {
		date d1 = new date();
		assertEquals(d1.nextDate(30, 6, 1921), "The next date is: 1-7-1921");
	}

	@Test
	public void test19() {
		date d1 = new date();
		assertEquals(d1.nextDate(31, 6, 1921), "Invalid date!");
	}

}


package EXP3;

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

import org.junit.jupiter.api.Test;

class WC_BVT {

	@Test
	public void test1() {
		date d1 = new date();
		assertEquals(d1.nextDate(1, 1, 1821), "The next date is: 2-1-1821");
	}

	@Test
	public void test2() {
```

```
    date d1 = new date();
    assertEquals(d1.nextDate(1, 1, 1822), "The next date is: 2-1-1822");
}

@Test
public void test3() {
    date d1 = new date();
    assertEquals(d1.nextDate(1, 1, 1921), "The next date is: 2-1-1921");
}

@Test
public void test4() {
    date d1 = new date();
    assertEquals(d1.nextDate(1, 1, 2020), "The next date is: 2-1-2020");
}

@Test
public void test5() {
    date d1 = new date();
    assertEquals(d1.nextDate(1, 1, 2021), "The next date is: 2-1-2021");

}

@Test
public void test6() {
    date d1 = new date();
    assertEquals(d1.nextDate(2, 1, 1821), "The next date is: 3-1-1821");
}

@Test
public void test7() {
    date d1 = new date();
    assertEquals(d1.nextDate(2, 1, 1822), "The next date is: 3-1-1822");
}

@Test
public void test8() {
    date d1 = new date();
    assertEquals(d1.nextDate(2, 1, 1921), "The next date is: 3-1-1921");
}

@Test
public void test9() {
    date d1 = new date();
    assertEquals(d1.nextDate(2, 1, 2020), "The next date is: 3-1-2020");
}

@Test
public void test10() {
    date d1 = new date();
    assertEquals(d1.nextDate(2, 1, 2021), "The next date is: 3-1-2021");
}

@Test
public void test11() {
    date d1 = new date();
    assertEquals(d1.nextDate(6, 1, 1821), "The next date is: 7-1-1821");
}

@Test
```

```
public void test12() {
    date d1 = new date();
    assertEquals(d1.nextDate(6, 1, 1822), "The next date is: 7-1-1822");
}

@Test
public void test13() {
    date d1 = new date();
    assertEquals(d1.nextDate(6, 1, 1921), "The next date is: 7-1-1921");
}

@Test
public void test14() {
    date d1 = new date();
    assertEquals(d1.nextDate(6, 1, 2020), "The next date is: 7-1-2020");
}

@Test
public void test15() {
    date d1 = new date();
    assertEquals(d1.nextDate(6, 1, 2021), "The next date is: 7-1-2021");
}

@Test
public void test16() {

    date d1 = new date();
    assertEquals(d1.nextDate(30, 1, 1821), "The next date is: 31-1-1821");
}

@Test
public void test17() {
    date d1 = new date();
    assertEquals(d1.nextDate(30, 1, 1822), "The next date is: 31-1-1822");
}

@Test
public void test18() {
    date d1 = new date();
    assertEquals(d1.nextDate(30, 1, 1921), "The next date is: 31-1-1921");
}

@Test
public void test19() {
    date d1 = new date();
    assertEquals(d1.nextDate(30, 1, 2020), "The next date is: 31-1-2020");
}

@Test
public void test20() {
    date d1 = new date();
    assertEquals(d1.nextDate(30, 1, 2021), "The next date is: 31-1-2021");
}

@Test
public void test21() {
    date d1 = new date();
    assertEquals(d1.nextDate(31, 1, 1821), "The next date is: 1-2-1821");
}
```

```
@Test
public void test22() {
date d1 = new date();
assertEquals(d1.nextDate(31, 1, 1822), "The next date is: 1-2-1822");
}

@Test
public void test23() {
date d1 = new date();
assertEquals(d1.nextDate(31, 1, 1921), "The next date is: 1-2-1921");
}

@Test
public void test24() {
date d1 = new date();
assertEquals(d1.nextDate(31, 1, 2020), "The next date is: 1-2-2020");
}

@Test
public void test25() {
date d1 = new date();
assertEquals(d1.nextDate(31, 1, 2021), "The next date is: 1-2-2021");
}

}
```

EXP 4----------Equivalence Class Partitioning for the NextDate Function

```
import java.util.*;

public class date {
public String nextDate(int d, int m, int y) {
int nd, nm, ny;
if (d > 31 || d < 1 || m > 12 || m < 1 || y < 1821 || y > 2021) {
return ("Invalid date!");
} else if (m == 2 || m == 4 || m == 6 || m == 9 || m == 11) {
if (d == 31)
return ("Invalid date!");
else if (m == 2) {
if (checkLeapYear(y)) {
if (d > 29) {
return ("Invalid date!");
}
if (d == 29) {
nd = 1;
nm = 3;
} else {
nd = ++d;
nm = 2;
}
} else {
if (d > 28) {
return ("Invalid date!");
}
if (d == 28) {
nd = 1;
nm = 3;
```

□□□□□}

□□□□□else {
□□□□□□nd = ++d;
□□□□□□nm = 2;
□□□□□}
□□□□}
□□□□ny = y;
□□□} else {
□□□□if (d == 30) {
□□□□□nd = 1;
□□□□□nm = ++m;
□□□□} else {
□□□□□nd = ++d;
□□□□□nm = m;
□□□□}
□□□□ny = y;
□□□}
□□} else {
□□□if (d == 31 && m != 12) {
□□□□nd = 1;
□□□□nm = ++m;
□□□□ny = y;
□□□} else if (d == 31 && m == 12) {
□□□□nd = 1;
□□□□nm = 1;
□□□□ny = ++y;
□□□} else {
□□□□nd = ++d;
□□□□nm = m;
□□□□ny = y;
□□□}
□□}
□□return ("The next date is: " + nd + "-" + nm + "-" + ny);
□}

□public static boolean checkLeapYear(int year) {
□□if (year % 400 == 0)
□□□return true;
□□else if (year % 100 == 0)
□□□return false;
□□else if (year % 4 == 0)
□□□return true;
□□else
□□□return false;
□}
}

```
package EXP4;

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

import org.junit.jupiter.api.Test;

class SN_ECT {

□@Test
□public void test1() {
□date d1 = new date();
```

```
assertEquals(d1.nextDate(14, 6, 2000), "The next date is: 15-6-2000");
}

}


package EXP4;

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

import org.junit.jupiter.api.Test;

class SN_ECT {

@Test
public void test1() {
date d1 = new date();
assertEquals(d1.nextDate(14, 6, 2000), "The next date is: 15-6-2000");
}

}


package EXP4;

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

import org.junit.jupiter.api.Test;

class SR_ECT {

@Test
public void test1() {
date d1 = new date();
assertEquals(d1.nextDate(15, 6, 2001), "The next date is: 16-6-2001");
}

@Test
public void test2() {
date d1 = new date();
assertEquals(d1.nextDate(0, 6, 1822), "Invalid date!");
}

@Test
public void test3() {
date d1 = new date();
assertEquals(d1.nextDate(15, 13, 1921), "Invalid date!");
}

@Test
public void test4() {
date d1 = new date();
assertEquals(d1.nextDate(15, 6, 2028), "Invalid date!");
}

@Test
public void test5() {
date d1 = new date();
assertEquals(d1.nextDate(0, 13, 2021), "Invalid date!");
}
```

```
	@Test
	public void test6() {
		date d1 = new date();
		assertEquals(d1.nextDate(41, 1, 1785), "Invalid date!");
	}

	@Test
	public void test7() {
		date d1 = new date();
		assertEquals(d1.nextDate(5, 15, 2112), "Invalid date!");
	}

	@Test
	public void test8() {
		date d1 = new date();
		assertEquals(d1.nextDate(46, 19, 1512), "Invalid date!");
	}

}
```

```
package EXP4;

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

import org.junit.jupiter.api.Test;

class WN_ECT {

	@Test
	public void test1() {
		date d1 = new date();
		assertEquals(d1.nextDate(14, 6, 2000), "The next date is: 15-6-2000");
	}

}
```

```
package EXP4;

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

import org.junit.jupiter.api.Test;

class WR_ECT {

	@Test
	public void test1() {
		date d1 = new date();
		assertEquals(d1.nextDate(15, 6, 2001), "The next date is: 16-6-2001");
	}

	@Test
	public void test2() {
		date d1 = new date();
		assertEquals(d1.nextDate(0, 6, 1822), "Invalid date!");
	}

	@Test
```

```
	public void test3() {
		date d1 = new date();
		assertEquals(d1.nextDate(15, 13, 1921), "Invalid date!");
	}

	@Test
	public void test4() {
		date d1 = new date();
		assertEquals(d1.nextDate(15, 6, 2028), "Invalid date!");
	}

}
```

EXP5----------The Triangle Problem White box testing

```
package EXP5;

import java.util.*;

public class triangle {
	public String check(int a, int b, int c) {
		while (true) {
			if (a >= 1 && a <= 200 && b >= 1 && b <= 200 && c >= 1 && c <= 200) {
				if ((a < b + c) && (b < a + c) && (c < b + a)) {
					if (a == b && b == c)
						return ("Given dimensions form an equilateral triangle!");
					else if (a == b || b == c || c == a)
						return ("Given dimensions form an isosceles triangle!");
					else
						return ("Given dimensions form a scalene triangle!");
				} else {
					return ("Given dimensions do not form a triangle!");
				}
			} else {
				return ("Enter a valid input!");
			}
		}
	}
}

package EXP5;

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

import org.junit.jupiter.api.Test;

class triangleTest {

	@Test
	public void test1() {
		triangle t1 = new triangle();
		assertEquals(t1.check(1, 2, 3), "Given dimensions do not form a triangle!");
	}

	@Test
	public void test2() {
		triangle t1 = new triangle();
		assertEquals(t1.check(2, 2, 2), "Given dimensions form an equilateral triangle!");
	}
```

```
@Test
public void test3() {
  triangle t1 = new triangle();
  assertEquals(t1.check(2, 2, 3), "Given dimensions form an isosceles triangle!");
}

@Test
public void test4() {
  triangle t1 = new triangle();
  assertEquals(t1.check(4, 5, 6), "Given dimensions form a scalene triangle!");
}

@Test
public void test5() {
  triangle t1 = new triangle();
  assertEquals(t1.check(-4, 5, 6), "Enter a valid input!");
}

@Test
public void test6() {
  triangle t1 = new triangle();
  assertEquals(t1.check(4, 5, 4), "Given dimensions form an isosceles triangle!");
}

@Test
public void test7() {
  triangle t1 = new triangle();
  assertEquals(t1.check(5, 4, 4), "Given dimensions form an isosceles triangle!");
}

@Test
public void test8() {
  triangle t1 = new triangle();
  assertEquals(t1.check(7, 4, 2), "Given dimensions do not form a triangle!");
}

@Test
public void test9() {
  triangle t1 = new triangle();
  assertEquals(t1.check(4, 7, 2), "Given dimensions do not form a triangle!");
}

@Test
public void test10() {
  triangle t1 = new triangle();
  assertEquals(t1.check(4, -5, 6), "Enter a valid input!");
}

@Test
public void test11() {
  triangle t1 = new triangle();
  assertEquals(t1.check(4, 5, -6), "Enter a valid input!");
}

@Test
public void test12() {
  triangle t1 = new triangle();
  assertEquals(t1.check(4, 205, -6), "Enter a valid input!");
}
```

```
@Test
public void test13() {
triangle t1 = new triangle();
assertEquals(t1.check(204, 205, 209), "Enter a valid input!");
}

@Test
public void test14() {
triangle t1 = new triangle();
assertEquals(t1.check(5, 5, 209), "Enter a valid input!");
}
}
```

EXP6----------The NextDate Function whitebox testion

```
package EXP6;


public class date {
public String nextDate(int d, int m, int y) {
int nd, nm, ny;
if (d > 31 || d < 1 || m > 12 || m < 1 || y < 1821 || y > 2021) {
return ("Invalid date!");
} else if (m == 2 || m == 4 || m == 6 || m == 9 || m == 11) {
if (d == 31) {
return ("Invalid date!");
} else if (m == 2) {
if (checkLeapYear(y)) {
if (d > 29) {
return ("Invalid date!");
}
if (d == 29) {
nd = 1;
nm = 3;
} else {
nd = ++d;
nm = 2;
}
} else {
if (d > 28) {
return ("Invalid date!");
}
if (d == 28) {
nd = 1;
nm = 3;
} else {
nd = ++d;
nm = 2;
}

}
ny = y;
} else {
if (d == 30) {
nd = 1;
nm = ++m;
} else {
nd = ++d;
nm = m;
}
```

```
        ny = y;
    }
  } else {
    if (d == 31) {
      if (m != 12) {
        nd = 1;
        nm = ++m;
        ny = y;
      } else {
        nd = 1;
        nm = 1;
        ny = ++y;
      }
    } else {
      nd = ++d;
      nm = m;
      ny = y;
    }
  }
  return ("The next date is: " + nd + "-" + nm + "-" + ny);
 }

 public static boolean checkLeapYear(int year) {
  if (year % 400 == 0)
    return true;
  else if (year % 100 == 0)
    return false;
  else if (year % 4 == 0)
    return true;
  else
    return false;
 }
}


package EXP6;

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

import org.junit.jupiter.api.Test;

import EXP6.date;

class dateECT {

 @Test
 public void test1() {
  date d1 = new date();
  assertEquals(d1.nextDate(29, 2, 2016), "The next date is: 1-3-2016");
 }

 @Test
 public void test2() {
  date d1 = new date();
  assertEquals(d1.nextDate(30, 2, 2016), "Invalid date!");
 }

 @Test
 public void test3() {
  date d1 = new date();
```

```
    assertEquals(d1.nextDate(29, 2, 2017), "Invalid date!");
}

@Test
public void test4() {
    date d1 = new date();
    assertEquals(d1.nextDate(31, 12, 2017), "The next date is: 1-1-2018");
}

@Test
public void test5() {
    date d1 = new date();
    assertEquals(d1.nextDate(2, 1, 2017), "The next date is: 3-1-2017");
}

@Test
public void test6() {
    date d1 = new date();
    assertEquals(d1.nextDate(31, 10, 2017), "The next date is: 1-11-2017");
}

@Test
public void test7() {
    date d1 = new date();
    assertEquals(d1.nextDate(20, 2, 2016), "The next date is: 21-2-2016");
}

@Test
public void test8() {
    date d1 = new date();
    assertEquals(d1.nextDate(28, 2, 2017), "The next date is: 1-3-2017");
}

@Test
public void test9() {
    date d1 = new date();
    assertEquals(d1.nextDate(31, 4, 1921), "Invalid date!");
}

@Test
public void test10() {
    date d1 = new date();
    assertEquals(d1.nextDate(15, 2, 1900), "The next date is: 16-2-1900");
}

@Test
public void test11() {
    date d1 = new date();
    assertEquals(d1.nextDate(15, 2, 2000), "The next date is: 16-2-2000");
}

@Test
public void test12() {
    date d1 = new date();
    assertEquals(d1.nextDate(32, 1, 2000), "Invalid date!");
}

@Test
public void test13() {
    date d1 = new date();
```

```
    assertEquals(d1.nextDate(30, 4, 2000), "The next date is: 1-5-2000");
  }

  @Test
  public void test14() {
    date d1 = new date();
    assertEquals(d1.nextDate(15, 4, 2000), "The next date is: 16-4-2000");
  }

  @Test
  public void test15() {
    date d1 = new date();
    assertEquals(d1.nextDate(0, 7, 2000), "Invalid date!");
  }

  @Test
  public void test16() {
    date d1 = new date();
    assertEquals(d1.nextDate(32, 7, 2000), "Invalid date!");
  }

  @Test
  public void test17() {
    date d1 = new date();
    assertEquals(d1.nextDate(15, 0, 2000), "Invalid date!");
  }

  @Test
  public void test18() {
    date d1 = new date();
    assertEquals(d1.nextDate(15, 15, 2000), "Invalid date!");
  }

  @Test
  public void test19() {
    date d1 = new date();
    assertEquals(d1.nextDate(15, 7, 1000), "Invalid date!");
  }

  @Test
  public void test20() {
    date d1 = new date();
    assertEquals(d1.nextDate(15, 7, 3000), "Invalid date!");
  }

  @Test
  public void test21() {
    date d1 = new date();
    assertEquals(d1.nextDate(15, 2, 2000), "The next date is: 16-2-2000");
  }

  @Test
  public void test22() {
    date d1 = new date();
    assertEquals(d1.nextDate(15, 4, 2000), "The next date is: 16-4-2000");
  }

  @Test
  public void test23() {
    date d1 = new date();
```

assertEquals(d1.nextDate(15, 6, 2000), "The next date is: 16-6-2000");
}

@Test
public void test24() {
date d1 = new date();
assertEquals(d1.nextDate(15, 9, 2000), "The next date is: 16-9-2000");
}

@Test
public void test25() {
date d1 = new date();
assertEquals(d1.nextDate(15, 11, 2000), "The next date is: 16-11-2000");
}

}

EXP8----------Using Selenium Web driver, automate any web page using Java Script

package EXP8;

import org.openqa.selenium.By;
import org.openqa.selenium.chrome.ChromeDriver;
public class pro8 {
public static void main(String[] args)throws InterruptedException {
System.setProperty("webdriver.chrome.driver","C:\\My_projects_works\\eclipse_workbench\\ST Lab\\chromedrive
ChromeDriver d = new ChromeDriver();
d.manage().window().maximize();
d.get("C:\\My_projects_works\\eclipse_workbench\\ST Lab\\login.html");
d.findElement(By.name("username")).sendKeys("bharath");
d.findElement(By.name("password")).sendKeys("bharath");
d.findElement(By.name("submit")).click();
}
}

```
<!DOCTYPE html>
<html>
<head>
   <title>Login Page</title>
</head>
<body>
   <h2>Login Form</h2>

   <div id="successfulLogin" style="display: none;">
      <h3>Login Successful!</h3>
   </div>

   <form id="loginForm">
      <label for="username">Username:</label><br>
      <input type="text" id="username" name="username" required><br>
      <label for="password">Password:</label><br>
      <input type="password" id="password" name="password" required><br>
      <input name="submit"  type="submit" value="Login">
   </form>

   <script>
      const form = document.getElementById('loginForm');
      const successDiv = document.getElementById('successfulLogin');
```

```
      form.addEventListener('submit', (event) => {
        event.preventDefault();

        const username = document.getElementById('username').value;
        const password = document.getElementById('password').value;

        if (username === 'bharath' && password === 'bharath') {
          successDiv.style.display = 'block';
          form.style.display = 'none';
        } else {
          alert('Login failed. Please check your username and password.');
        }
      });
    </script>
</body>
</html>
```

EXP9----------List the total number of objects present on a web page

```
package EXP9;

import org.openqa.selenium.By;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import java.util.List;
public class pro9 {
public static void main(String[] args)throws InterruptedException {
System.setProperty("webdriver.chrome.driver","C:\\My_projects_works\\eclipse_workbench\\ST Lab\\chromedriver
ChromeDriver d=new ChromeDriver();
d.get("C:\\My_projects_works\\eclipse_workbench\\ST Lab\\login.html");
//d.get("https://en.wikipedia.org/wiki/Bangalore");
d.manage().window().maximize();
List<WebElement> links=d.findElements(By.xpath("//form"));
//List<WebElement> links=d.findElements(By.xpath("//a"));
int linkc=links.size();
System.out.println("Total no of link count on webpage=" + linkc);
List<WebElement> allElement=d.findElements(By.xpath("//*"));
int elementscount=allElement.size();
System.out.println("Total no of all element on webpage=" + elementscount);
}
}
```

EXP10----------Demonstrate URL and title check point

```
package EXP10;

import org.openqa.selenium.By;

import org.openqa.selenium.chrome.ChromeDriver;

public class exp10 {
public static void main(String args[]) {
  System.setProperty("webdriver.chrome.driver",
    "C:\\My_projects_works\\eclipse_workbench\\ST Lab\\chromedriver-win64\\chromedriver.exe");
  ChromeDriver d = new ChromeDriver();
  d.manage().window().maximize();
  // d.get("https://en.wikipedia.org/wiki/Wikipedia");
```

```
d.get("C:\\My_projects_works\\eclipse_workbench\\ST Lab\\login.html");
String url = d.getCurrentUrl();
System.out.println("Current URL: " + url);
// if (url.equals("https://en.wikipedia.org/wiki/Wikipedia"))
if (url.equals("file:///C:/My_projects_works/eclipse_workbench/ST%20Lab/login.html"))
System.out.println("URL matches!");
else
System.out.println("URL doesn't match.");

d.get("https://www.google.com");
String title = d.getTitle();
System.out.println("Current Title: " + title);
if (title.equals("Google"))
System.out.println("Title matches!");
else
System.out.println("Title doesn't match.");
}
}
```

EXP11----------Demonstrate selecting and deselecting option from multi select dropdown

```
package EXP11;

import java.util.List;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.Select;

public class exp11 {
public static void main(String[] args) throws InterruptedException {
System.setProperty("webdriver.chrome.driver",
"C:\\My_projects_works\\eclipse_workbench\\ST Lab\\chromedriver-win64\\chromedriver.exe");
ChromeDriver driver = new ChromeDriver();
driver.manage().window().maximize();
driver.get("C:\\My_projects_works\\eclipse_workbench\\ST Lab\\branch.html");
Select select = new Select(driver.findElement(By.id("depts")));
System.out.println("The multiselect options are: ");
List<WebElement> options = select.getOptions();
for (WebElement option : options)
System.out.println(option.getText());
System.out.println("Is the selected element a multiselect element?: " + select.isMultiple());
if (select.isMultiple()) {
System.out.println("Selecting option ECE using its index.");
select.selectByIndex(2);
Thread.sleep(4000);

System.out.println("Selecting option ISE using its value.");
select.selectByValue("ise");
Thread.sleep(4000);

System.out.println("Selecting option CSE using its visible text.");
select.selectByVisibleText("CSE");
Thread.sleep(4000);

System.out.println("The selected options are: ");
options = select.getAllSelectedOptions();
for (WebElement option : options)
```

```
        System.out.println(option.getText());

    System.out.println("Deselecting option ECE using its index.");
    select.deselectByIndex(2);
    Thread.sleep(4000);

    System.out.println("Deselecting option ISE using its value.");
    select.deselectByValue("ise");
    Thread.sleep(4000);

    System.out.println("The selected values after deselecting some options are: ");
    options = select.getAllSelectedOptions();
    for (WebElement option : options)
        System.out.println(option.getText());

    System.out.println("Deselecting all options.");
    select.deselectAll();
  }
 }
}


<h1>
Select and Deselect Departments
</h1>
<form>
<select multiple name="depts" id="depts">
<option value="cse">CSE</option>
<option value="ise">ISE</option>
<option value="ece">ECE</option>
<option value="eee">EEE</option>
</select>
</form>


EXP12----------------------------------------------Demonstrate Synchronization.

EXPLICIT

package EXP12;

import java.time.Duration;

import org.openqa.selenium.By;
import org.openqa.selenium.TimeoutException;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;

public class exp12_explicit {
 public static void main(String[] args) {
  System.setProperty("webdriver.chrome.driver",
      "C:\\My_projects_works\\eclipse_workbench\\ST Lab\\chromedriver-win64\\chromedriver.exe");

  ChromeDriver driver = new ChromeDriver();
  driver.manage().window().maximize();
  String eText = "WELCOME"; // Expected Text
  String aText_1 = "";
  String aText_2 = "";
```

```
  driver.get("C:\\My_projects_works\\eclipse_workbench\\ST Lab\\welcome.html");
  WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(5));

  try {
    WebElement text_1 = wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("welcome")));
    aText_1 = text_1.getText();
    if (aText_1.equals(eText))
      System.out.println("Test 1 Passed using Explicit Wait");
  } catch (TimeoutException e) {
    System.out.println("Test 1 Failed using Explicit Wait");
  }
  try {
    WebElement text_2 = wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("abcd")));
    aText_2 = text_2.getText();
    if (aText_2.equals(eText))
      System.out.println("Test 2 Passed using Explicit Wait");
  } catch (TimeoutException e) {
    System.out.println("Test 2 Failed using Explicit Wait");
  }
 }
}
```

IMPLICIT

```
package EXP12;

import java.util.concurrent.TimeUnit;
import org.openqa.selenium.NoSuchElementException;
import org.openqa.selenium.By;
import org.openqa.selenium.chrome.ChromeDriver;

public class exp12_implicit {
 public static void main(String[] args) {
  System.setProperty("webdriver.chrome.driver",
    "C:\\My_projects_works\\eclipse_workbench\\ST Lab\\chromedriver-win64\\chromedriver.exe");
  ChromeDriver driver = new ChromeDriver();
  driver.manage().window().maximize();
  String eText = "WELCOME";
  String aText_1 = "";
  String aText_2 = "";

  driver.get("C:\\My_projects_works\\eclipse_workbench\\ST Lab\\welcome.html");
  driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

  try {
    aText_1 = driver.findElement(By.id("welcome")).getText();
    if (aText_1.equals(eText))
      System.out.println("Test 1 Passed using Implicit Wait");
  } catch (NoSuchElementException e) {
    System.out.println("Test 1 Failed using Implicit Wait");
  }
  try {
    aText_2 = driver.findElement(By.id("abcd")).getText();
    if (aText_2.equals(eText))
      System.out.println("Test 2 Passed using Implicit Wait");
  } catch (NoSuchElementException e) {
    System.out.println("Test 2 Failed using Implicit Wait");
```

□□}
□}
}

<h1 id="welcome">WELCOME</h1>
<p><i><b>New Horizon College of Engineering is an Autonomous college affiliated to Visvesvaraya Technological Uni
tion (AICTE) & University Grants Commission (UGC). It is accredited by NAAC with ‘A’ grade & National Board of Accre
the heart of the IT capital of India, Bangalore. The college campus is situated in the IT corridor of Bangalore surrounc
<h2 id="abcd">WELCOME</h1>