DAY 7

1) Check if character is a Digit

```
public class DigitChecker {
   public static void main(String[] args) {
      char ch = '3';
      System.out.println(Character.isDigit(ch));
   }
}
```

Output

True

2) Method accepts integer and tries to change value

```
public class PassByValueExample {
   public static void changeValue(int x) {
      x = 75;
   }

   public static void main(String[] args) {
      int num = 25;
      System.out.println("Before: " + num);
      changeValue(num);
      System.out.println("After: " + num);
   }
}
```

Output

Before: 25

After: 25

3) Modify object field

```
class Box {
   int length;
}

public class ModifyObject {
   public static void changeLength(Box b) {
      b.length = 70;
   }

   public static void main(String[] args) {
      Box box = new Box();
      box.length = 25;
      System.out.println("Before: " + box.length);
      changeLength(box);
      System.out.println("After: " + box.length);
   }
}
```

Output

Before: 25

After: 70

4) Pass object and modify field

```
class Person {
   String name;
}

public class ModifyPerson {
   public static void changeName(Person p) {
      p.name = "Charlie";
   }

   public static void main(String[] args) {
      Person person = new Person();
      person.name = "Bob";
      System.out.println("Before: " + person.name);
      changeName(person);
      System.out.println("After: " + person.name);
   }
}
```

Output
Before: Bob
After: Charlie

5) Update marks of Student

```
class Student {
   String name;
   int marks;
}

public class UpdateStudentMarks {
   public static void updateMarks(Student s) {
      s.marks = 85;
   }

   public static void main(String[] args) {
      Student st = new Student();
      st.name = "Kiran";
      st.marks = 70;
      System.out.println("Before: " + st.marks);
      updateMarks(st);
      System.out.println("After: " + st.marks);
   }
}
```

Output
Before: 70
After: 85

6) Create thread by extending Thread class (1-5)

```
class MyThread extends Thread {
   public void run() {
      for (int i = 10; i <= 15; i++)
         System.out.println(i);
   }
}
```

```
public class ExtendThreadExample {
   public static void main(String[] args) {
      new MyThread().start();
   }
}
```

Output

10
11
12
13
14
15

7) Create thread by implementing Runnable (print thread name)

```
class MyRunnable implements Runnable {
   public void run() {
      System.out.println("Running on: " + Thread.currentThread().getName());
   }
}

public class RunnableExample {
   public static void main(String[] args) {
      new Thread(new MyRunnable(), "CustomThread-1").start();
      new Thread(new MyRunnable(), "CustomThread-2").start();
   }
}
```

Output

Running on: CustomThread-2
Running on: CustomThread-1

8) Two threads printing different messages 5 times

```
class MessageThread extends Thread {
   String msg;

   MessageThread(String m) {
      msg = m;
   }

   public void run() {
      for (int i = 0; i < 3; i++)
         System.out.println(msg);
   }
}

public class TwoMessageThreads {
   public static void main(String[] args) {
      new MessageThread("First").start();
      new MessageThread("Second").start();
   }
}
```

Output

First

Second
First
Second
First
Second

9) Demonstrate Thread.sleep()

```
public class SleepDemo {
   public static void main(String[] args) throws InterruptedException {
      for (int i = 5; i <= 7; i++) {
         System.out.println(i);
         Thread.sleep(1000);
      }
   }
}
```

Output
5

6

7

10) Thread.yield() usage

```
class YieldThread extends Thread {
   public void run() {
      for (int i = 0; i < 2; i++) {
         System.out.println(getName());
         Thread.yield();
      }
   }
}

public class YieldDemo {
   public static void main(String[] args) {
      YieldThread t1 = new YieldThread();
      YieldThread t2 = new YieldThread();
      t1.setName("Thread-A");
      t2.setName("Thread-B");
      t1.start();
      t2.start();
   }
}
```

Output
Thread-A
Thread-B
Thread-A
Thread-B

11) Two threads: even and odd numbers

```
class EvenPrinter extends Thread {
   public void run() {
      for (int i = 20; i <= 26; i += 2)
         System.out.println("Even: " + i);
```

```
    }
}

class OddPrinter extends Thread {
    public void run() {
        for (int i = 21; i < 27; i += 2)
            System.out.println("Odd: " + i);
    }
}

public class EvenOddThreads {
    public static void main(String[] args) {
        new EvenPrinter().start();
        new OddPrinter().start();
    }
}
```

Output Even: 20
Odd: 21
Even: 22
Odd: 23
Even: 24
Odd: 25
Even: 26

12) Three threads with different priorities

```
class PriorityThread extends Thread {
    public void run() {
        System.out.println(getName() + " Priority: " + getPriority());
    }
}

public class PriorityDemo {
    public static void main(String[] args) {
        PriorityThread threadA = new PriorityThread();
        PriorityThread threadB = new PriorityThread();
        PriorityThread threadC = new PriorityThread();
        threadA.setName("Thread-A");
        threadB.setName("Thread-B");
        threadC.setName("Thread-C");
        threadA.setPriority(Thread.MIN_PRIORITY);
        threadC.setPriority(Thread.MAX_PRIORITY);
        threadA.start();
        threadB.start();
        threadC.start();
    }
}
```

Output
Thread-A Priority: 1
Thread-B Priority: 5
Thread-C Priority: 10

13) Thread.join() usage

```
class JoinThread extends Thread {
   public void run() {
      for (int i = 5; i <= 7; i++)
         System.out.println(getName() + " " + i);
   }
}

public class JoinDemo {
   public static void main(String[] args) throws InterruptedException {
      JoinThread t1 = new JoinThread();
      t1.start();
      t1.join();
      System.out.println("Main thread finished after t1");
   }
}
```

Output

Thread-0 5
Thread-0 6
Thread-0 7
Main thread finished after t1