Loops.

Loops implement repetitive computation, a.k.a. iteration.

Java loop statements:

- `while`
- `do-while`
- `for`

1. Start with the `while`-loop. Syntax

   ```java
   while (condition) {
     //body of loop
     First_Statement;
     ...
     Last_Statement;
   }
   ```

Several logical loop organizations

- counting loops – know exactly how many times to repeat a set of actions
  - usually done with a use of a counter variable
  - counter is initialized before the loop starts executing
  - counter is updated after each iteration of the loop

**Example:** print numbers from 1 to m.

```java
count = 1; // initialize the counter variable
while (count <= m) {
    System.out.print(count);
    count = count + 1; // same as count++
}
```

**Question:**
What will happen if m = -5?
- condition controlled loops – repeat while a certain condition is true
  - sentinel-controlled loops – stop repeating when a certain value is encountered
    **Example:** Let user input numbers until user enters 0.
    0 is the *sentinel* in this case.

    ```java
    aNum = SavitchIn.readLineInt();
    while ( aNum != 0 ) {
        System.out.print(aNum);
        aNum = SavitchIn.readLineInt();
    }
    ```

    **Question:** how would you modify this segment to also compute the sum and the product of all entered numbers?

  - boolean flag controlled loops – stop when a boolean variable that reflects a certain state is false
    **Example:** figure out when this loop is going to terminate.

    ```java
    int prevPos = 0;
    int currPos = 1;
    String str = SavitchIn.readLine();
    boolean foundDouble = false; // boolean flag
    int len = str.length();
    while (!foundDouble && currPos <= len - 1) {
        if (str.charAt(prevPos) == str.charAt(currPos))
            foundDouble = true;
        prevPos++;
        currPos++;
    }
    ```

Programming and Debugging Pitfalls:
- infinite loops
- off by one
2. do-while loop: similar to while-loop, but condition is checked at the end of each iteration. The body of the loop is going to be executed at least once.

```
do
{
    //body of loop
    First_Statement;
    ...
    Last_Statement;
} while(condition)
```

3. for –loops – especially useful for counting loops. Syntax:

```
for (init-statements; loop-condition; update-statements)
{
    //body of loop
    First_Statement;
    ...
    Last_Statement;
}
```

**Example:** count down from 9 to 0

```java
for( count = 9; count >= 0; count-- )
{
    System.out.print("T = "+ count);
    System.out.println(" and counting");
}
System.out.println("Blast off!");
```
4. Nested loops:

When one loop is placed within the body of another, the entire construct is called “nested loops”.

Example:

```java
for (m = 1; m <= 10; m++)
{
    count = 1; // initialize the counter variable
    while ( count <= m ) {
        System.out.print(count);
        count = count + 1; // same as count++
    }
    System.out.println(\n        ***\n    );
}
```

Practice problems.

1. Define a static method that takes a string and returns the reverse of that string, e.g. given “Walter” it would return “retlaW”.

2. Define a static method that takes a string and a character and returns the number of occurrences of the character inside that string.