Assignment 2.4: Loops

Writing Programs that Use the Terminal

0. Writing to the Terminal

In this project, we will be sending our answers to the terminal for the user to see. To write numbers and text to the Terminal, use System.out.print() and System.out.println(). Println writes the code to the terminal and adds a line break at the end, where as print does not add the line break. Here are some examples:

```
//Print text
//System.out.println("What is the meaning of life?");

//Print the value of a variable:
System.out.println(count);

//Print the value of a variable and text:
System.out.println("You have played this game " + count + " times!");
```

Make sure you find out the difference between these two sets of statements:

`System.out.print("Hello");` `System.out.print("Hello");`	`System.out.println("Hello");` `System.out.println("Hello");`

1. Reading User Input from the Terminal

In this project, the user will be able to interact with your program by typing in responses in the Terminal. We will use the Scanner class to accomplish this. Note that reading user input is not part of the AP Exam and will never be tested, but it will be important to understand to make many projects work during the course.

In the assignment, a Scanner object called input has been declared and initialized already for you. Here are some examples of how to use it:

```
int n = input.nextInt(); //reads an int from terminal and stores it
double val = input.nextDouble(); //same, but for decimal values
String ans = input.next(); //stores text
```

It is often important to include a prompt to tell the user what information you expect. Using print instead of println means the question can be answered on the same line as the question. For example:

```
System.out.print("What is your name? ");
```

```
String name = input.next();
System.out.println(“Hello “ + name);
```

2. Checking if the Value of Strings

If you want to see if a String variable has a certain value, you cannot use the == operator. We will discuss this more in the future Strings unit. For now, you need to use the following technique:

RIGHT	WRONG

`String ans = input.nextLine();`
`if (ans.equals(“yes”)) {`
`  System.out.println(“YESSS!”);`
`} else if (ans.equals(“no”))`
`  System.out.println(“NOOOO!”);`
`}`
	`String ans = input.nextLine();`
	`if (ans == “yes”) {`
	`  System.out.println(“YESSS!”);`
	`} else if (ans == “no”)`
	`  System.out.println(“NOOOO!”);`
	`}`

3. Generating Random Numbers

The Math class includes the method Math.random() which returns a number between 0 (inclusive) and 1 (exclusive). To generate a number in a different range, you need to use multiplication and addition to shift this range. If you want to create random integers, you will need to use casting operators to remove the decimal components.

An Example Program That Uses the Terminal

Here is an example method that interacts with the user in the terminal to convert between centimeters and inches:

```
    public void inchesToCM() {
        double cm;
        int feet, inches, remainder;
        double CM_PER_INCH = 2.54;
        int IN_PER_FOOT = 12;
        // prompt the user and get the value
        System.out.print("Exactly how many cm? ");
        cm = in.nextDouble();
        // convert and output the result
        inches = (int) (cm / CM_PER_INCH);
        feet = inches / IN_PER_FOOT;
        remainder = inches % IN_PER_FOOT;
        System.out.print(cm + “cm = “ + feet + “ft, “ + inches +
          “in”);
    }
}
```

For Loops

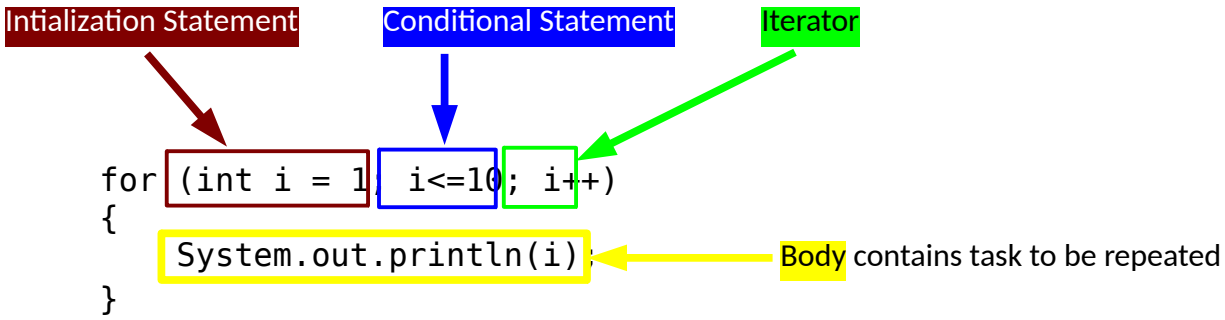
A for loop is used to repeat a set of instructions. Here is an example. Please read it carefully and try to figure out what it does before reading on.

Example 1:

```
for (int i = 1; i<=10; i++)
{
     System.out.println(i);
}
```

Check your answer at the end

Parts of a For Loop



Initialization Statement

The Initialization Statement creates a variable to keep track of how many times we have repeated the loop, and what value we will start counting from. In this case, we call this variable "i" and we start counting from 1. The initialization statement only happens the first time we start the loop, not every time we go through the loop.

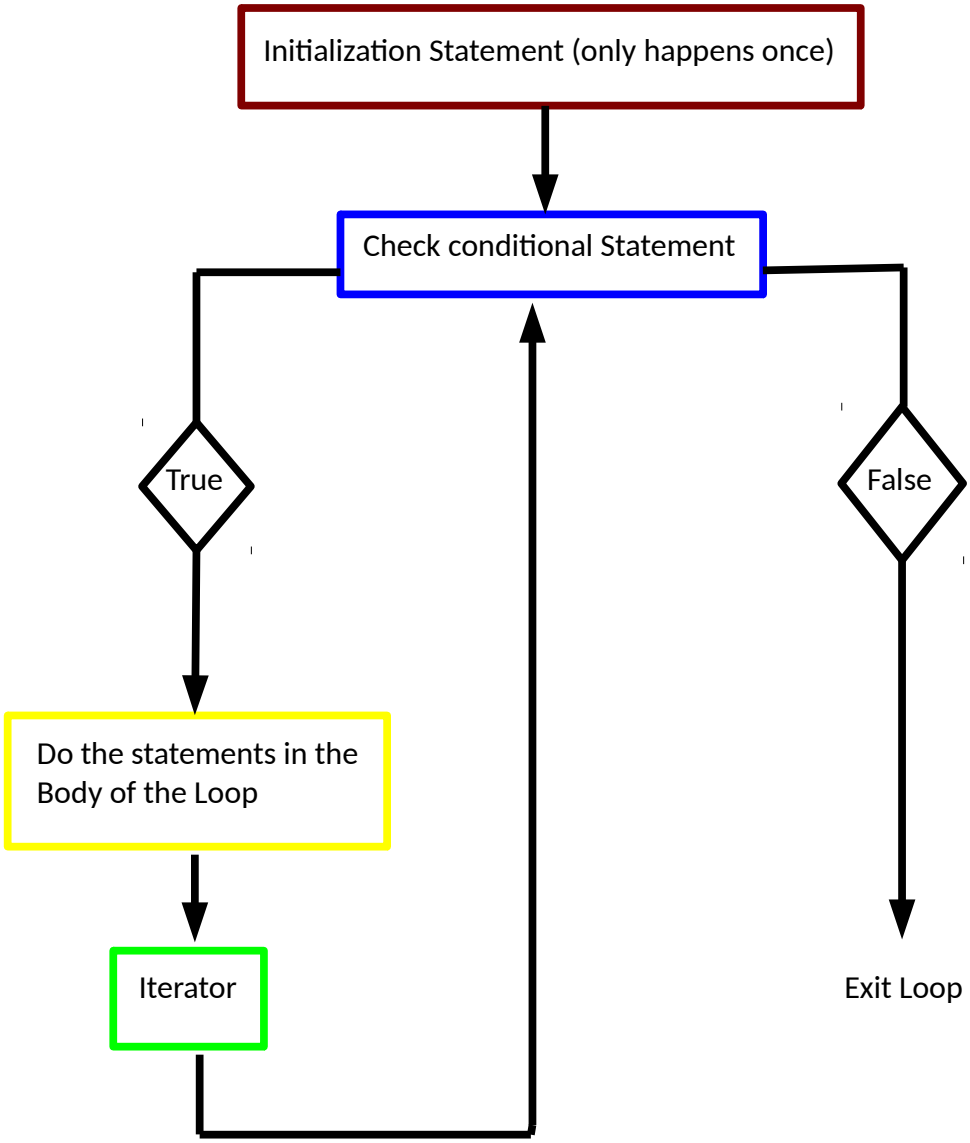
Conditional Statement

The Conditional Statement is a boolean (true or false) expression that determines if we repeat the loop or not. The statement evaluates to true, the task will be repeated another time. If it is false, the loop will stop. The Conditional Statement is checked every time it repeats the loop, including the FIRST time through the loop. So, it is possible that the body of the loop doesn't even get to run once.

Iterator

The Iterator is how we are counting as we go through the loop. The code `i++` is short for "add 1 to i", which means we are counting by ones. You could replace this with other statements, such as `i--` to count down by 1 or `i = i * 2` to double i every time.

How A Loop Works in Detail



Using this flowchart, we can see that the output from our example loop will be: 12345678910

While Loops

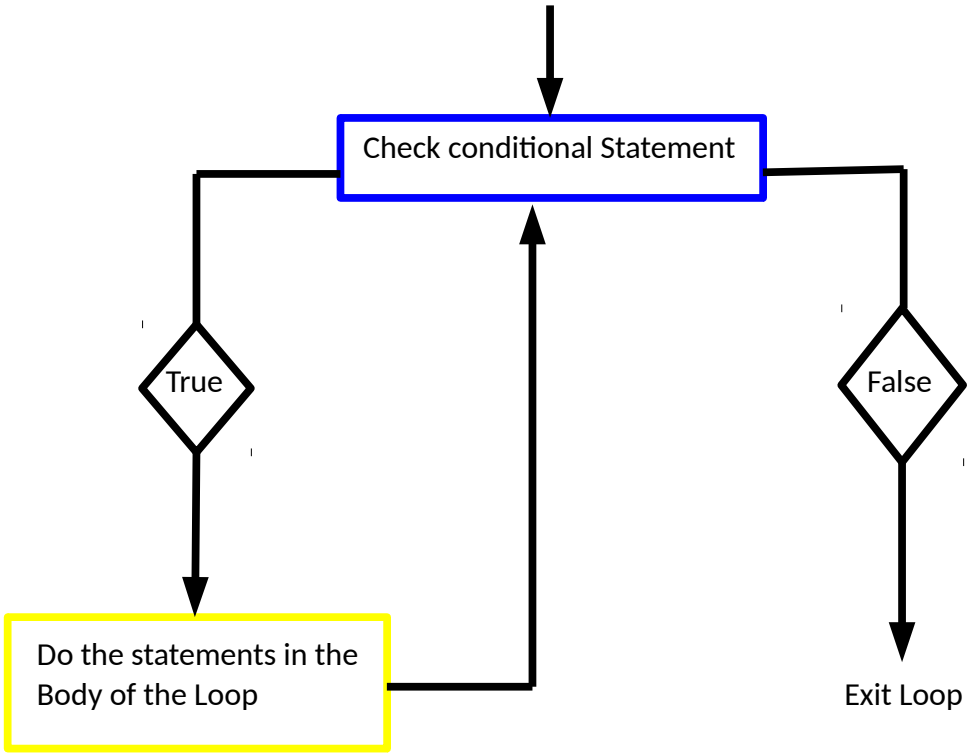
While loops are the most used loop in Java. It is the simplest and most versatile loop.

```
while (true)
{
        System.out.print("!");
}
```

Conditional Statement

Body contains task to be repeated

As long as the conditional statement is true, the loop will repeat the statements in the body of the loop. The flow chart for this kind of loop looks like this:



If the conditional statement is false at the beginning of the loop, then the loop will **not** execute the statements in the body of the loop even once.

All for loops can also be written as while loops. For example, this for loop:

```
for (int count = 0; count < 11; count++)
{
     System.out.println("Count is: " + count);
}
```

Can be written as the following while loop:

```
int count = 0;
while (count < 11)
{
     System.out.println("Count is: " + count);
     count++;
}
```

Take some time to trace through this loop and understand how the two loops work the same.

As you can see, **for loops** are simpler for repeating a task a set number of times. However, **while loops** can be like "forever if" loops in scratch. They will continue to repeat something until the condition is false.

Here is an example of a while loop that does not repeat a task a set number of times, but keeps doing the task until some condition becomes false.

```
1    public void whileTest()
2    {
3        Scanner input = new Scanner(System.in);
4
5        String answer = "Mike";
6
7        while (answer.equals("Mike"))
8        {
9            System.out.print("What's your name, Mike? ");
10           answer = input.next();
11       }
12   System.out.print("Hello, " + answer);
13   }
```

A sample of the output of the program is as follows:

```
What's your name, Mike?  Mike
What's your name, Mike?  Mike
What's your name, Mike?  Mike
What's your name, Mike?  Bob
Hello, Bob
```

Think about the answer to these question about the above loop. If you don't know the answers, ask for help!

On line 5, we initialize the variable answer to "Mike". What would have happened if we set it to "Bob" ?

How does Java know when to stop the loop and move on to the next statements?

Nested Loops

A nested loop is where you have a loop as a statement in the body of another loop. For example:

```
for (int k = 0; k <= 3; k++) {
      for (int i = 0; i < 4; i++) {
            System.out.print(“>”);
      }
      System.out.println(“”);
}
```

In this case, the inside loop, which produces a row of four greater-than signs like this: >>>> is repeated 3 times, each time, adding a line break at the end. The result will be:

```
>>>>
>>>>
>>>>
```

EXERCISE 1:

Consider this for loop:

```
for (i = 100; i >= 10; i--)
{
     System.out.println("i is: " + i);
}
```

Rewrite this as a while loop.

EXERCISE 2:

What is the output of this loop?

```
for (i = 1; i <= 10; i++)
{
     for (j = 1 ; j <= i; j++)
     {
          System.out.print(j + " ");
     }
     System.out.println("");
}
```

EXERCISE 3:

Write class RunningTally that repeatedly asks the user to enter an integer until the users gives a negative integer. At that point, the program prints out the total of all the numbers that the user entered (not including the negative). For example:

```
Let's add some numbers! [Type a negative number to quit]

Add:  4
Add:  6
Add:  2
Add:  -1

Your total is 12.
```

Note: Although the total is displayed at the end, it is important to update the total every time the user enters a new number.

EXERCISE 4:

Write class Bar that asks a user for a String char and an int x. The program then prints out a row of chars that is as long as x. For example, your program would look like this:

```
What character do you want your bar made of? *
How long do you want your bar? 7

Here's your bar!

*******
```

EXERCISE 5:

Write class Triangle that asks a user for a String char and an int x. Then the program prints out a right triangle of chars that is as tall as x. For example:

```
What character do you want your triangle made of? *
How tall do you want your triangle? 5

Here's your triangle!

*
**
***
****
*****
******
```

This will require a nested for loop (a for loop inside a for loop).

EXERCISE 6:

Cut and paste your old D6 program into this project. Change your code so that it is contained within a loop. Change it so it runs like the following output:

```
How many D6's do you want to roll? 3

You rolled: 2 5 4
Total: 11

Again? [y,n]  y

How many D6's do you want to roll? 5

You rolled: 1 2 1 3 4
Total: 11

Again? [y,n]  n
Good-bye!
```

EXERCISE 7:

Write class GuessingGame that simulates a the "Guess What Number I'm Thinking Of" game. In this game, the computer picks a random number between 1 and 100. Then, the method will begin a loop, asking the user to guess the number. After each guess, the method will tell the user whether the guess was too high or too low, and then repeat the process. If the user guesses correctly, the loop will stop and the method will then display a congratulatory message and tell you how many tries it took you to guess the correct answer. Sample output is as follows:

```
I've picked a random number between 1 and 100.  Try to guess it!
What is your guess?  50
Too high ...
What is your guess?  25
Too high ...
What is your guesss?  12
Too low ...
What is your guess?  16
You've guessed my number! Good job!  It only took you 4 tries.
```