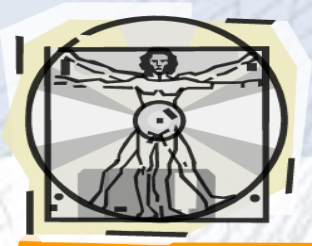
Computer Programming

Strings

Özgür Koray ŞAHİNGÖZ
Prof.Dr.

Biruni University
Computer Engineering Department

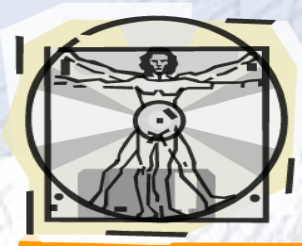


Motivations

Suppose that you need to print a string (e.g., "Welcome to Java!") a hundred times. It would be tedious to have to write the following statement a hundred times:

```
System.out.println("Welcome to Java!");
```

So, how do you solve this problem?

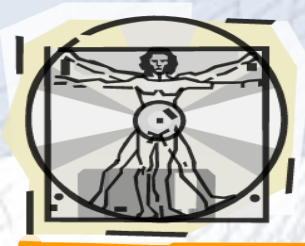


Opening Problem

Problem:

100 times

```
System.out.println("Welcome to Java!");
System.out.println("Welcome to Java!");
System.out.println("Welcome to Java!");
System.out.println("Welcome to Java!");
System.out.println("Welcome to Java!");
System.out.println("Welcome to Java!");

...

...

...
System.out.println("Welcome to Java!");
System.out.println("Welcome to Java!");
System.out.println("Welcome to Java!");
```

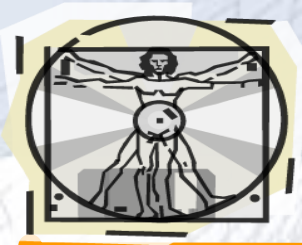


The while Statement

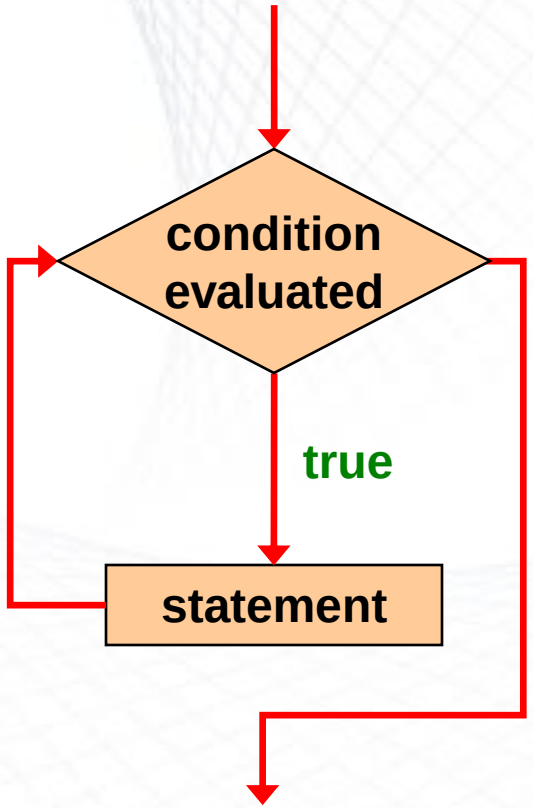
- A *while statement* has the following syntax:

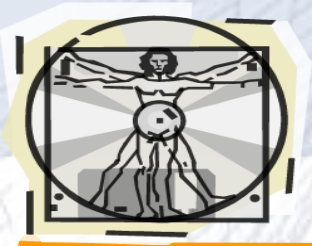
```
while ( condition )
   statement;
```

- If the **`condition`** is true, the **`statement`** is executed
- Then the condition is evaluated again, and if it is still true, the statement is executed again
- The statement is executed repeatedly until the condition becomes false



Logic of a while Loop



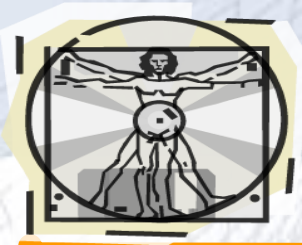


The while Statement

- An example of a while statement:

```
int count = 1;
while (count <= 5) {
   System.out.println(count);
   count++;
}
```

- If the condition of a `while` loop is false initially, the statement is never executed
- Therefore, the body of a `while` loop will execute zero or more times

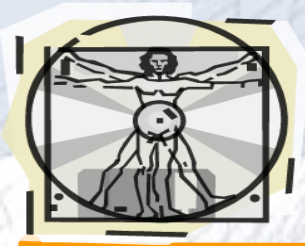


Trace while Loop

```
int count = 0;
```

Initialize count

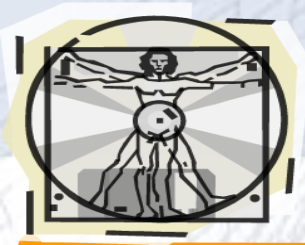
```
while (count < 2) {
  System.out.println("Welcome to Java!");
  count++;
}
```



Trace while Loop, cont.

(count < 2) is true

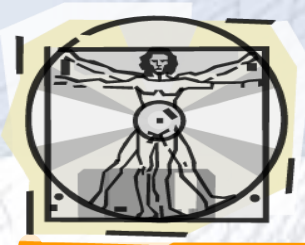
```
int count = 0;

while (count < 2) {
  System.out.println("Welcome to Java!");
  count++;
}
```



Trace while Loop, cont.

```
int count = 0;

while (count < 2) {
  System.out.println("Welcome to Java!");
  count++;
}
```

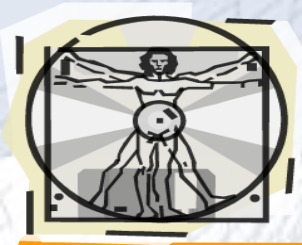
Print Welcome to Java



Trace while Loop, cont.

```
int count = 0;

while (count < 2) {

  System.out.println("Welcome to Jav    );

  count++;

}
```

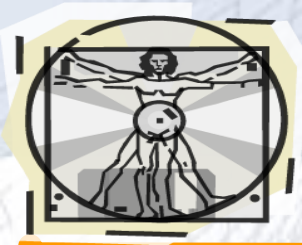
Increase count by 1
count is 1 now



Trace while Loop, cont.

```
int count = 0;

while (count < 2) {

  System.out.println("Welcome to Java!");

  count++;

}
```

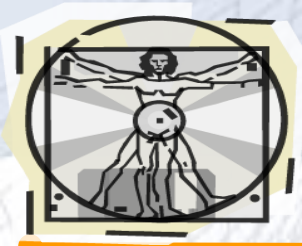
(count < 2) is still true since count is 1



Trace while Loop, cont.

```
int count = 0;

while (count < 2) {
  System.out.println("Welcome to Java!");
  count++;
}
```

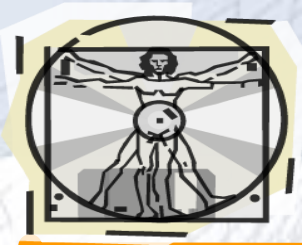
Print Welcome to Java



Trace while Loop, cont.

```
int count = 0;

while (count < 2) {

  System.out.println("Welcome to Java");

  count++;

}
```

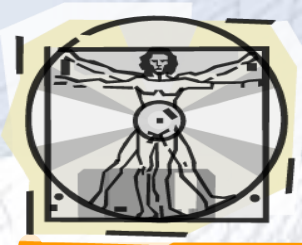
Increase count by 1
count is 2 now



Trace while Loop, cont.

```
int count = 0;

while (count < 2) {

  System.out.println("Welcome to Java!");

  count++;

}
```

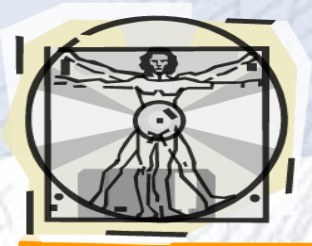
(count < 2) is false since count is 2 now



Trace while Loop

```
int count = 0;
while (count < 2) {
  System.out.println("Welcome to Java!
  count++;
}
```

The loop exits. Execute the next statement after the loop.

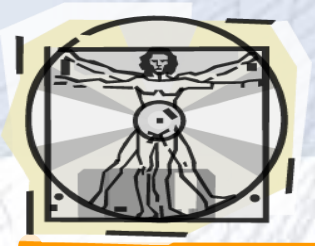


Example

- This program will find the summation of numbers from 1 to 10.

Solution

```
class whileLoopDemo {
    public static void main(String args[])    {
        int x = 1, sum = 0;
        while (x <= 10) {      // Exit when x becomes greater than 4
            sum = sum + x;
            x++;
        }
        System.out.println("Summation: " + sum);
    }
}
```

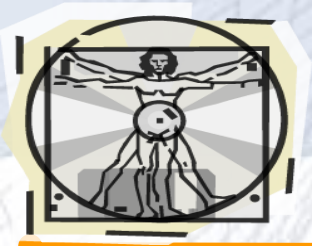


Problem: Repeat Addition Until Correct

Recall that Listing 3.1 AdditionQuiz.java gives a program that prompts the user to enter an answer for a question on addition of two single digits. Using a loop, you can now rewrite the program to let the user enter a new answer until it is correct.

IMPORTANT NOTE: If you cannot run the buttons, see .

RepeatAdditionQuiz **Run**

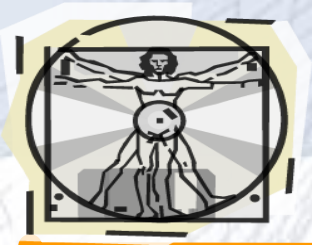


Problem: Guessing Numbers

Write a program that randomly generates an integer between 0 and 100, inclusive. The program prompts the user to enter a number continuously until the number matches the randomly generated number. For each user input, the program tells the user whether the input is too low or too high, so the user can choose the next input intelligently. Here is a sample run:

GuessNumberOneTim Run

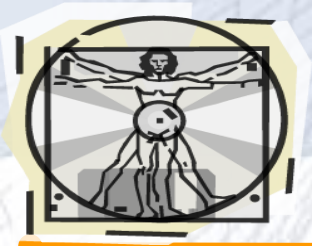
GuessNumber Run



Problem: An Advanced Math Learning Tool

The Math subtraction learning tool program generates just one question for each run. You can use a loop to generate questions repeatedly. This example gives a program that generates five questions and reports the number of the correct answers after a student answers all five questions.

SubtractionQuizLoop **Run**

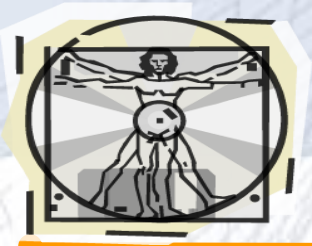


Ending a Loop with a Sentinel Value

Often the number of times a loop is executed is not predetermined. You may use an input value to signify the end of the loop. Such a value is known as a *sentinel value*.

Write a program that reads and calculates the sum of an unspecified number of integers. The input 0 signifies the end of the input.

SentinelValue **Run**

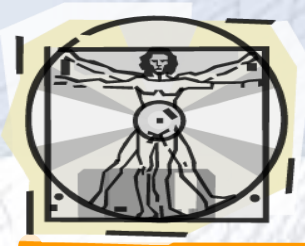


Ending a Loop with a Sentinel Value

Often the number of times a loop is executed is not predetermined. You may use an input value to signify the end of the loop. Such a value is known as a *sentinel value*.

Write a program that reads and calculates the sum of an unspecified number of integers. The input 0 signifies the end of the input.

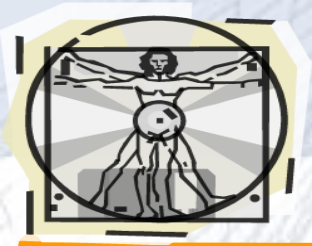
SentinelValue **Run**



Caution

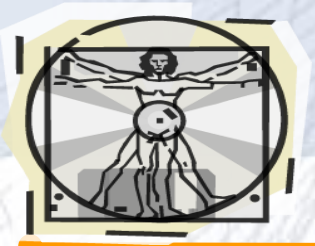
Don’t use floating-point values for equality checking in a loop control. Since floating-point values are approximations for some values, using them could result in imprecise counter values and inaccurate results. Consider the following code for computing 1 + 0.9 + 0.8 + ... + 0.1:

```
double item = 1; double sum = 0;
while (item != 0) { // No guarantee item will be 0
  sum += item;
  item -= 0.1;
}
System.out.println(sum);
```



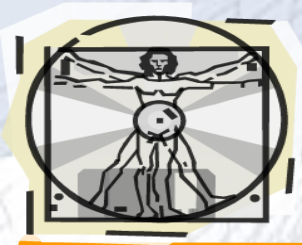
Example

- Read 10 Values
- And calculate the average of them



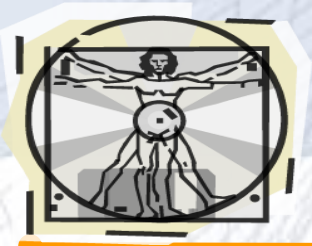
Average

```
import java.util.Scanner;

public class WelcomeJava {

public static void main(String[] args) {

int sum = 0, value, count = 0;
double average;

Scanner scan = new Scanner(System.in);
System.out.print("");

System.out.println("Enter 10 Integer Values: ");

while (count != 10){  // 10 times looping
    count++;
    System.out.print("Enter an integer\t: ");
    value = scan.nextInt();
    sum += value;
    System.out.println("The sum so far is  \t:" + sum);
    }
    System.out.println();
    average = (double)sum / count;
    System.out.println("The average is \t :" + average);
  }
}
```



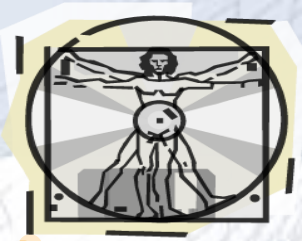
Sample Run

```
Enter 10 Integer Values
Enter an integer: 25
The sum so far is 25
Enter an integer: 164
The sum so far is 189
Enter an integer: -14
The sum so far is 175
Enter an integer: 84
The sum so far is 259
Enter an integer: 12
The sum so far is 271
Enter an integer: -35
The sum so far is 236
Enter an integer: 12
The sum so far is 248
Enter an integer: 10
The sum so far is 258
Enter an integer: -8
The sum so far is 250
Enter an integer: 6
The sum so far is 256

The average is 25.6
```



Example

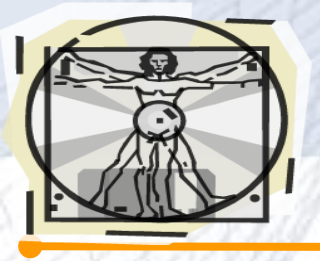
- Read integer values
- If it is finished enter 0 to exit
- `And calculate the average of them`



Average

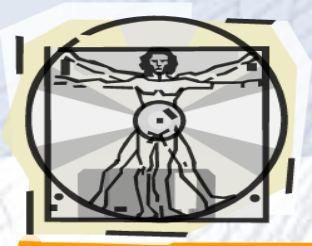
```
import java.util.Scanner;

public class Average{
public static void main(String[] args)   {
    int sum = 0, value=-1, count = 0;
    double average;

    Scanner scan = new Scanner(System.in);
```

```
while (value != 0){  // sentinel value of 0 to terminate loop
            System.out.print("Enter an integer (0 to quit): ");
            value = scan.nextInt();
            count++;
            sum += value;
            System.out.println("The sum so far is " + sum);
     }
    System.out.println();
     if (count == 0)
       System.out.println("No values were entered.");
     else {
       average = (double)sum / count;
       System.out.println("The average is " + average);
     }
   }
}
```



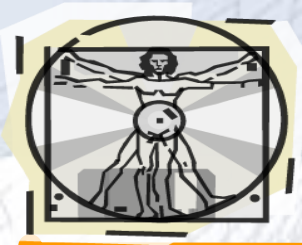
Sample Run

```
Enter an integer (0 to quit): 25
The sum so far is 25
Enter an integer (0 to quit): 164
The sum so far is 189
Enter an integer (0 to quit): -14
The sum so far is 175
Enter an integer (0 to quit): 84
The sum so far is 259
Enter an integer (0 to quit): 12
The sum so far is 271
Enter an integer (0 to quit): -35
The sum so far is 236
Enter an integer (0 to quit): 0

The average is 39.333
```



Input Validation

- A loop can also be used for *input validation*, making a program more *robust*
- It's generally a good idea to verify that input is valid (in whatever sense) when possible
- See `WinPercentage.java`

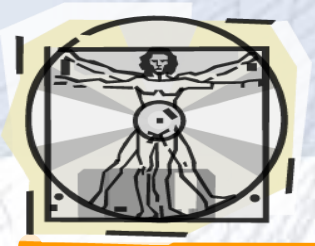


WinPercentage.java

```
public class WinPercentage  {
public static void main(String[] args)  {
      final int NUM_GAMES = 12;        int won;        double ratio;
      Scanner scan = new Scanner(System.in);

      System.out.print("Enter the number of games w
                       + NUM_GAMES + "): ");
      won = scan.nextInt();
      while (won < 0 || won > NUM_GAMES)       {
         System.out.print("Invalid input. Please re
         won = scan.nextInt();
      }

      ratio = (double)won / NUM_GAMES;
      System.out.println();
      System.out.println("Winning percentage: " + ratio);
   }
}
```

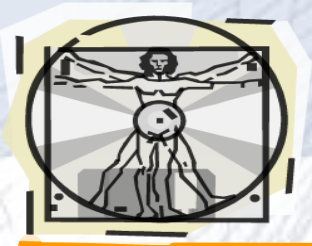
Sample Run

```
Enter the number of games won (0 to 12): -5
Invalid input. Please reenter: 13
Invalid input. Please reenter: 7

Winning percentage: 58%
```



Infinite Loops

- The body of a `while` loop eventually must make the condition false
- If not, it is called an *infinite loop*, which will execute until the user interrupts the program
- This is a common logical error
- You should always double check the logic of a program to ensure that your loops will terminate normally

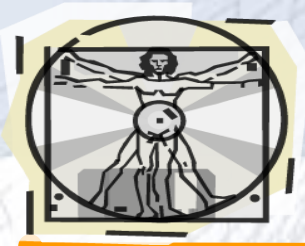


Infinite Loops

- An example of an infinite loop:

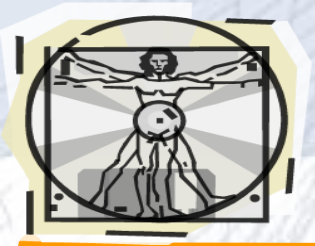
```
int count = 1;
while (count <= 25) {
    System.out.println(count);
    count = count - 1;
}
```

- This loop will continue executing until interrupted (Control-C) or until an underflow error occurs



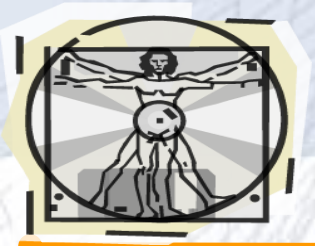
Nested Loops

- Similar to nested `if` statements, loops can be nested as well
- That is, the body of a loop can contain another loop
- For each iteration of the outer loop, the inner loop iterates completely
- See `PalindromeTester.java`



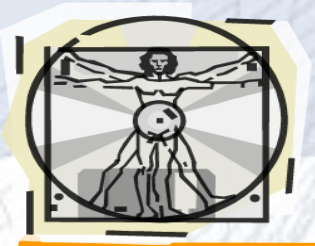
PalindromeTester.java

```
public class PalindromeTester{
public static void main(String[] args)
   {
      String str, another = "y";
      int left, right;

      Scanner scan = new Scanner(System.in);

      while (another.equalsIgnoreCase("y")){
         System.out.println("Enter a String");
         str = scan.nextLine();
         left = 0;
         right = str.length() - 1;

         while (str.charAt(left) == str.charAt(right)
                  && left < right)    {
            left++;
            right--;
         }

         System.out.println();

         if (left < right)
            System.out.println(" NOT a palindrome.");
         else
            System.out.println("That string IS a palindrome.");

         System.out.println();
         System.out.print("Test another palindrome (y/n)? ");
         another = scan.nextLine();
          }
        }
      }
```



Sample Run

```
Enter a potential palindrome:
radar

That string IS a palindrome.

Test another palindrome (y/n)? y
Enter a potential palindrome:
able was I ere I saw elba

That string IS a palindrome.

Test another palindrome (y/n)? y
Enter a potential palindrome:
abracadabra

That string is NOT a palindrome.

Test another palindrome (y/n)? n
```

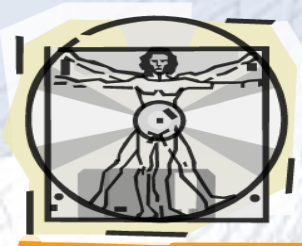


Quick Check

How many times will the string "Here" be printed?

```
count1 = 1;
while (count1 <= 10) {
   count2 = 1;
   while (count2 < 20)    {
      System.out.println("Here");
      count2++;
   }
   count1++;
}
```

10 * 19 = 190



Game Programming

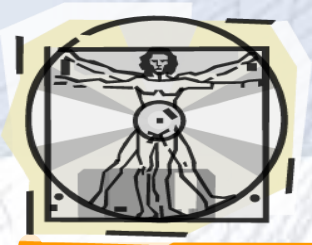
- Write a program
- which flips Coins 100 times
- and calculate the number of HEADs in the Gam

- Use
- **import java.Math.*;**
- **double** a = Math.random(); // Generates a value between 0..1

Game Programming-2

- Write a program
- which rolls a Dice
- and calculate the all number of values in the Game.

- Use
- **import java.Math.*;**
- **double** a = Math.random(); // Generates a value between 0..1



Game Programming - 3

- Let Computer to hold a value between 0..99
- As a user try to estimate it.
- Write the number of guess to the screen

```
I hold a number
Please find this value

User's Guess    : 50
Result   : Up
User's Guess    : 75
Result   : Up
User's Guess    : 82
Result   : Down
User's Guess    : 80
Result   : Ok

User Find the number in his 4 th guess
```