Java: Decisions and Loops

CAC 210

Amber Wagner

Decision Statements

```
// Statements that execute before the branches

if (expression) {
    // Statements that execute when expression is true (first branch)
}
else {
    // Statements that execute when expression is false (second branch)
}

// Statements that execute after the branches
```

Example

```
import java.util.Scanner;

public class InsuranceQuote {
  public static void main(String[] args) {
     Scanner scnr = new Scanner(System.in);
     int userAge;
     int insurancePrice;

     System.out.print("Enter age: ");
     userAge = scnr.nextInt();

     if (userAge < 25) {
        insurancePrice = 4800;
     }
     else {
        insurancePrice = 2200;
     }

     System.out.print("Annual price: $");
     System.out.println(insurancePrice);
  }
}
```

If-else-if

```
import java.util.Scanner;

public class MultIfElseAnniv {
   public static void main(String[] args) {
      Scanner scnr = new Scanner(System.in);
      int numYears;

      System.out.print("Enter number years married: ");
      numYears = scnr.nextInt();

      if (numYears == 1) {
         System.out.println("Your first year -- great!");
      }
      else if (numYears == 10) {
         System.out.println("A whole decade -- impressive.");
      }
      else if (numYears == 25) {
         System.out.println("Your silver anniversary -- enjoy.");
      }
      else if (numYears == 50) {
         System.out.println("Your golden anniversary -- amazing.");
      }
      else {
         System.out.println("Nothing special.");
      }
   }
}
```

You Write

- Write a program to determine if a year is a leap year. In order for a year to be a leap year, it must be divisible by four. However, if a year is divisible by 100, it must also be divisible by 400 in order to be a leap year.

Ternary Operator

- Shortcut for a simple question
- condition ? true : false
- Instead of writing…

```
int a = 10, b = 20, c;
if(a < b)
    c = a;
else
    c = b;
```

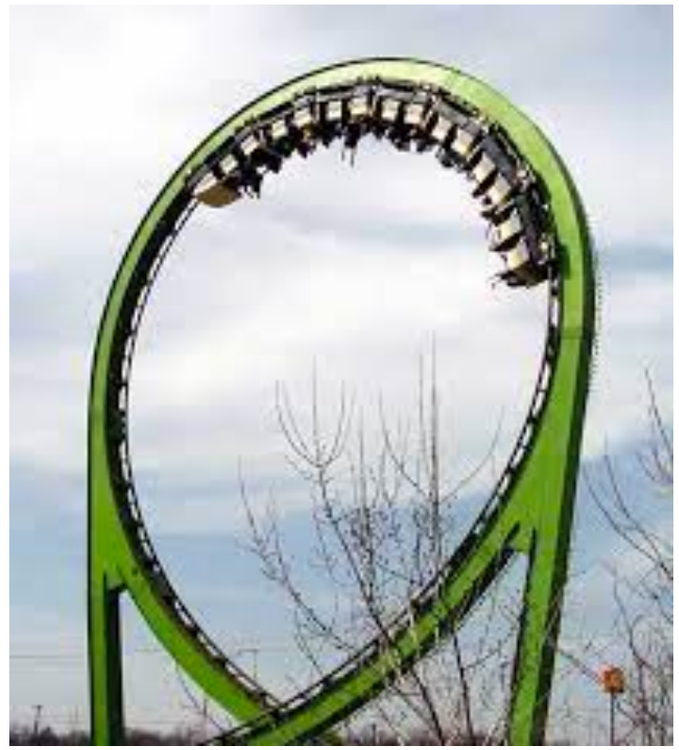
- We can write…

```
int a = 10, b = 20, c;
c = a < b ? a : b
```

Logical Operators

- And - &&
- Or - ||
- Not - !

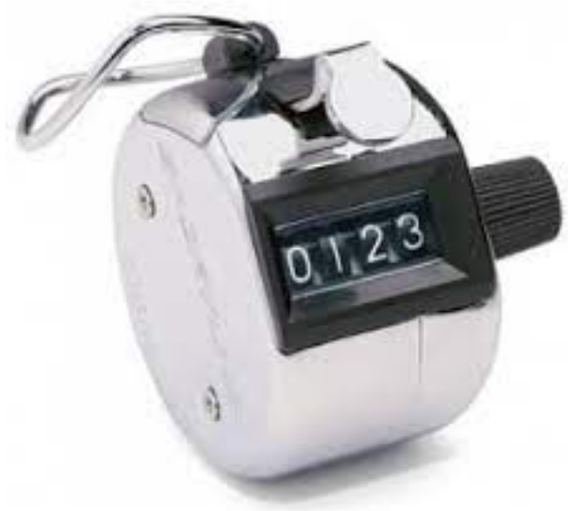
Loops



- For loops: definite loops
- For each loops
- While/Do While loops: indefinite loops

For Loops

- Definite loops
- They have a definite end
- Counting loops




```
for(start; stop; step) {

}
```

You Write

- Create an array of Strings called fruits.
- Populate it by asking the user to enter their five favorite fruits (hint: use a for loop).
- Print all of the fruits the user entered (hint: use a for loop)

For Each

- This loop is super cool!
- Let’s say I have that array of Strings…

```
for(String fruit : fruits) {
    System.out.println(fruit);
}
```

- What’s the difference? Why use one over the other?

While Loop

- Indefinite, why?

```
while (condition) {

}
```

- Why would I use this instead of a for loop?

Do…while

- A little different

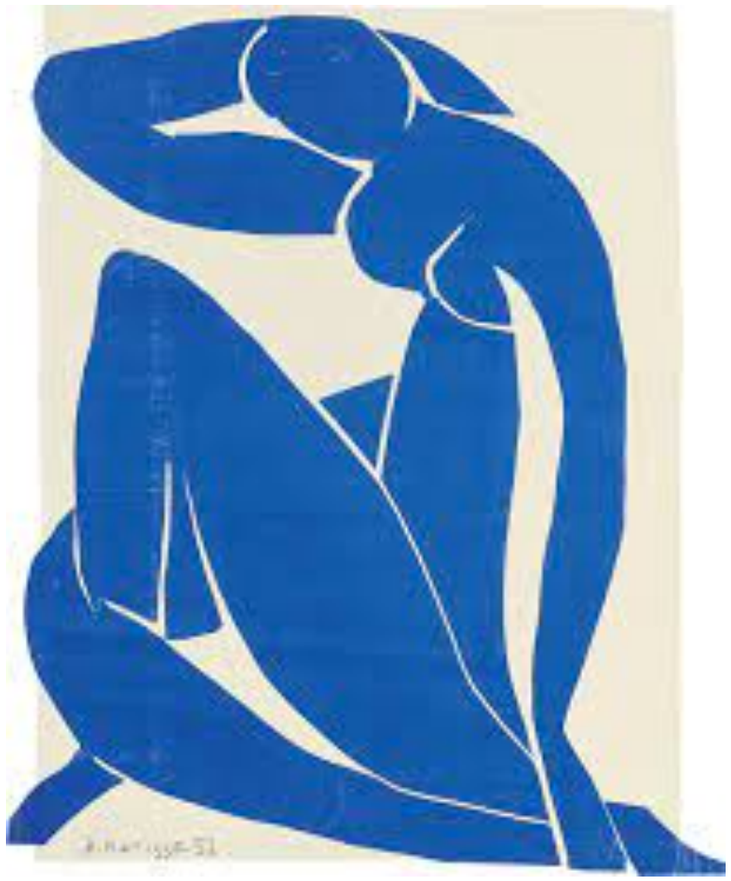
  do {

  } while (condition)

- What’s the difference?

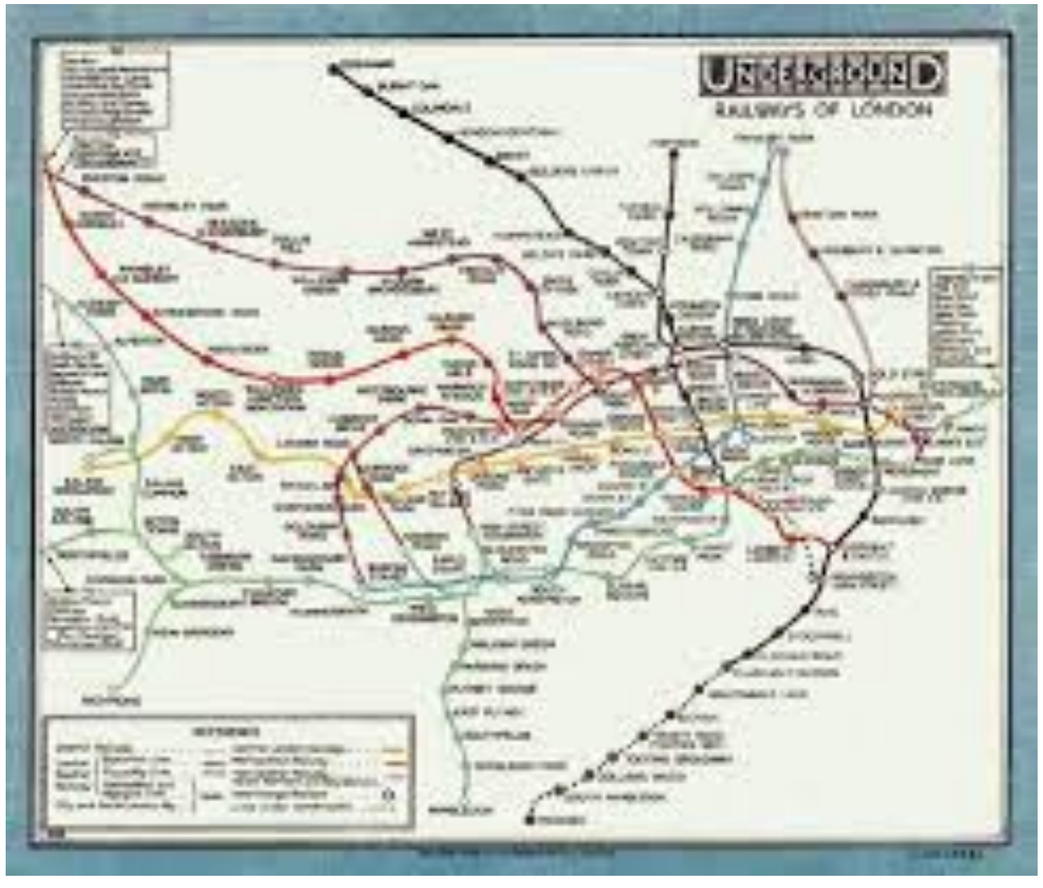
- Why would you use this over the traditional while?

ArrayList



- This is more like a Python list than an array is

- It is not a primitive type…ArrayList is a class that has something like an array on the backend, but allows programmers to use the list **abstraction**



ArrayList

- Must import the package

  import java.util.ArrayList;

- Declare a new ArrayList object

  ArrayList<Integer> numbers = new ArrayList<Integer>();

- Add elements to the ArrayList

  numbers.add(aNumber);

Why do you think this is here?

You Write

- Ask the user to enter positive numbers and store them in an ArrayList. When they enter a -1 stop reading numbers.
- Print the average
- Hint: use two different types of loops

Next Class…

- Finish Mini-Assignment #1
- Take a peek at Mini-Assignment #2 due Sept. 8 (decisions and loops)
- Read about Java Classes