Multiple If Statement

```
if (score >= 90.0)
        grade = 'A';
else if (score >= 80.0)
        grade = 'B';
else if (score >= 70.0)
         grade = 'C';
else if (score >= 60.0)
        grade = 'D';
else grade = 'F';
```

Logical Operators

Operator	Name
!	not
&&	and
||	or
^	exclusive or

Change Maker

Problem:

you have to give someone change

what coins do you give that person?

Requirements:

takes user input

displays the change breakdown as output

1. Understand and Define the Problem

ask user for input, US coins (quarter, dime, nickel, penny), max change: 99¢ display coin output ,

What's involved?

interview users. What are their expectations? What data do they need to access? write a requirements analysis report

Change Maker

2.Determine Input and Output

Typed input by user: amount of change requested (an integer between 1 and 99)

Printed output:

Number of quarters given

Number of dimes given

Number of nickels given

Number of pennies given

Switch Statements

```
switch (var) {
    case 0:         ...;
        break;
    case 1:         ...;
        break;
    case 2:         ...;
        break;
    default: ...;
    }
```

Packages

To make types easier to find and use, to avoid naming conflicts, and to control access, programmers bundle groups of related types into packages.

The types that are part of the Java platform are members of various packages that bundle classes by function: fundamental classes are in java.lang, classes for reading and writing (input and output) are in java.io, and so on.

You can put your types in packages too.

To create a package, you choose a name for the package and put a package statement with that name at the top of every source file that contains the types (e.g., classes, interfaces).

In file Circle.java:

```
package edu.claimAcademy.introToJava;

public class Circle {

        …

}
```

The String Type

The char type only represents one character.

To represent a string of characters, use the data type called String.

```
String message = "Welcome to Java";
```

String is a predefined class in the Java library just like the System class.

The String type is NOT a primitive type.

The String type is a reference type.

String variable = a reference variable, which points to an object storing the value or actual text

The String Type

Each character is stored at an index:

```
    String sentence = "A statement";
                       012345678910
```

The String class (from J2SE) has methods to process strings:

```
System.out.println("charAt(6) is " + sentence.charAt(6));
System.out.println(sentence.toUpperCase());
System.out.println(sentence.substring(0,7) + sentence.substring(14));
```

The String Type

There are no methods to change them once they have been created

any new assignment will assign a new

String to the old variable

```
    String word = "Steven";
    word = word.substring(0, 5);
```

the variable word is a reference to a new String that contains "Steve"

The String Type

Don’t use ‘==’ to compare Strings

it compares their memory addresses and not actual strings (character sequences)

Instead use the equals/1 method supplied by the String class

Iteration

We have 3 types of iterative statements

- a while loop
- a do … while loop
- a for loop

All 3 can be used to do similar things Which one should you use? a matter of individual preference/convenience

Intro to java

Iteration

```
do {
      // Loop body;
      Statement(s);
} while (loop-continuation-condition);
```

```
for (int i = 0; i < 100; i++){
      System.out.println( "Welcome to Java!");
      }
```

```
while (loop-continuation-condition) {
// loop-body;
Statement(s);
}
```

Intro to java

Iteration

Find multiple sums of integers:

- from 1 to 10,

- from 20 to 30,

- from 35 to 45,

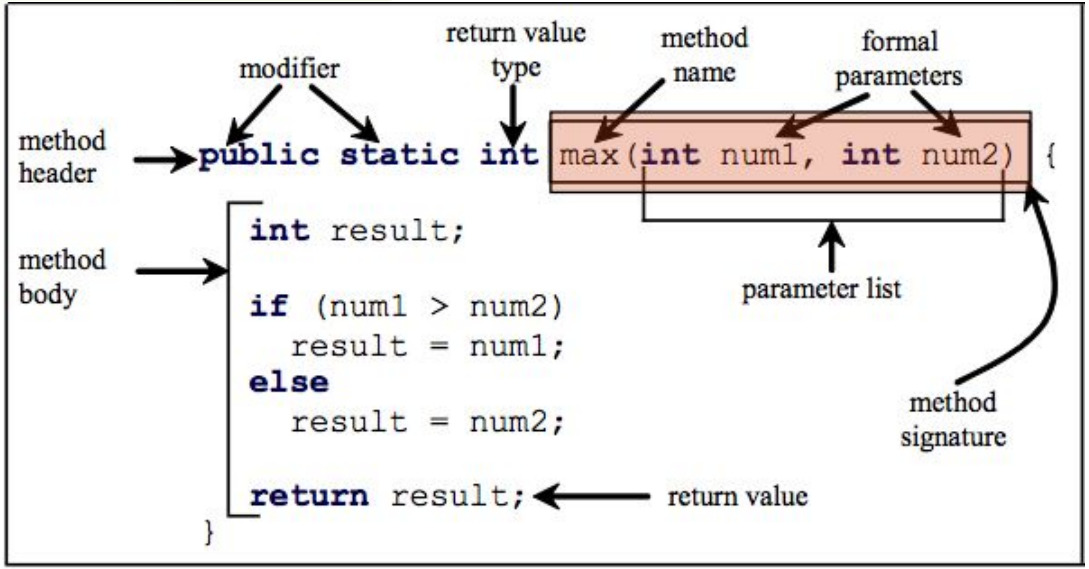
```
public static int sum(int i1, int i2) {
int sum = 0;
for (int i = i1; i <= i2; i++)
        sum += i;
return sum;
}
```

Intro to java

Methods

A method is a collection of statements that are grouped together to perform an operation.

Method signature is the combination of the method name and the parameter list



```
public static int max(int num1, int num2) {
   int result;

   if (num1 > num2)
     result = num1;
   else
     result = num2;

   return result;
}
```

Why write Methods

To shorten your programs

avoid writing identical code twice or more

To modularize your programs

fully tested methods can be trusted

To make your programs more:

readable, reusable

testable, debuggable

extensible, adaptable