Introduction to java

COMP1030
Lecture #5

Housekeeping

- Our goal - 100% pass rate in this course.
- Review the lecture slides ahead of time.
- Review the lecture slides after class with a study group.
- Repeat the lab at home 1-2 times.
- Take notes during class.
- Weekly optional tutorials take place every week on Wednesdays, from 3-4pm in room A132.

Review

- Pseudocode
- Flow Charts
- If-Else
- Nested If- Else
- While Statement
- Increment/Decrement Operators
- Relational Operators
- Boolean Operators

for Statement

for (*initialization*; *termination*; *increment*)
{
    statement(s)
}

for Statement

```
class ForDemo
{
        public static void main(String[] args)
        {
                for(int i=1; i<11; i++)
                {
                        System.out.println("Count is: " + i);
                }
        }
}
```

for Statement

The output of this program is:

```
Count is: 1
Count is: 2
Count is: 3
Count is: 4
Count is: 5
Count is: 6
Count is: 7
Count is: 8
Count is: 9
Count is: 10
```

do-while Statement

```
Do
{
        statement(s)
} while (expression);
```

The difference between do-while and while is that the do-while evaluates its expression at the bottom of the loop instead of the top. Therefore, the statements within the do block are always executed at least once.

do-while Statement

codingground
SIMPLY EASY CODING

Compile and Execute Java Online (JDK 1.8.0)

Fork Project

Execute | Share Source File STDIN

```java
public class HelloWorld{

    public static void main(String []args){
        int count = 1;
        do
        {
            System.out.println("count is: " + count);
            count++;
        }while (count<1);
    }
}
```

Result

```
$javac HelloWorld.java
$java -Xmx128M -Xms16M HelloWorld
count is: 1
```

switch Statement

Syntax

```
switch(variable) {
    case valueOne:
        //statements
        break;
    case valueTwo:
        //statements
        break;
    default: //optional
    //statements
}
```

switch Statement

Execute | Share | Source File | STDIN

```java
public class HelloWorld{

    public static void main(String []args)
    {
        int x = 1;
        switch (x)
        {
            case 1:
                System.out.println("one");
                break;
            case 2:
                System.out.println("two");
                break;
            case 3:
                System.out.println("three");
                break;
        }
    }
}
```

Result

```
$javac HelloWorld.java
$java -Xmx128M -Xms16M HelloWorld
one
```

switch Statement

Execute | Share | Source File | STDIN

```java
public class HelloWorld{

    public static void main(String []args)
    {
        int x = 1;
        switch (x)
        {
            case 1:
                System.out.println("one");
            case 2:
                System.out.println("two");
                break;
            case 3:
                System.out.println("three");
                break;
        }
    }
}
```

Result

```
$javac HelloWorld.java
$java -Xmx128M -Xms16M HelloWorld
one
two
```

switch Statement

Execute | Share | Source File | STDIN

```java
public class HelloWorld{

    public static void main(String []args)
    {
        int x = 1;
        switch (x)
        {
            case 1:
                System.out.println("one");
            case 2:
                System.out.println("two");
            case 3:
                System.out.println("three");
                break;
        }
    }
}
```

Result

```
$javac HelloWorld.java
$java -Xmx128M -Xms16M HelloWorld
one
two
three
```

switch Statement

The following rules apply to a **switch** statement –

- The variable used in a switch statement can only be integers, convertable integers (byte, short, char), strings and enums.
- You can have any number of case statements within a switch. Each case is followed by the value to be compared to and a colon.
- The value for a case must be the same data type as the variable in the switch and it must be a constant or a literal.
- When the variable being switched on is equal to a case, the statements following that case will execute until a *break* statement is reached.
- When a *break* statement is reached, the switch terminates, and the flow of control jumps to the next line following the switch statement.
- Not every case needs to contain a break. If no break appears, the flow of control will *fall through* to subsequent cases until a break is reached.
- A *switch* statement can have an optional default case, which must appear at the end of the switch. The default case can be used for performing a task when none of the cases is true. No break is needed in the default case.