Object Oriented Methodology Lab

Md. Mujahid Islam

Software Developer & Guest Lecturer

Control Statement

- Java’s program control statements can be put into the following categories: selection, iteration, and jump. *Selection* statements allow your program to choose different paths of execution based upon the outcome of an expression or the state of a variable. *Iteration* statements enable program execution to repeat one or more statements (that is, iteration statements form loops). *Jump* statements allow your program to execute in a nonlinear fashion. All of Java’s control statements are examined here.
- Java supports two selection statements: if and switch.

If Selection

if (*condition*) *statement1*;

else *statement2*;

Example :

```
if(a==b){
      System.out.println("Dhaka");
}else{
      System.out.println("Rajshahi");
}
```

Nested ifs

if (*condition*) *{*

    if (*condition*) *statement1*;

        if (*condition*) *statement1*;

        else *statement2*;

    }

else *statement2*;

```
double a=mScanner.nextDouble();

if(a>0){

        if(a<100){
                System.out.println("Ok");
        }else{
                System.out.println("Invalid Input");
        }

}else{
        System.out.println("Invalid Input");
}
```

The if-else-if Ladder

if(*condition*) *statement*;

else if(*condition*) *statement*;

else if(*condition*) *statement*;

.

.

.

else *statement*;

```
if(a==0){
	System.out.println("Equal to Zero");

}else if(a<0){
	System.out.println("Ok");

}
else if(a>0){
	System.out.println("Greater then Zero");

}
else{
	System.out.println("Invalid Input");
}
```

Switch

- The switch statement is Java's multiway branch statement. It provides an easy way to dispatch execution to different parts of your code based on the value of an expression.
- For versions of Java prior to JDK 7, *expression* must be of type byte, short, int, char, or an enumeration. Beginning with JDK 7, *expression* can also be of type String.
- Each value specified in the case statements must be a unique constant expression. Duplicate case values are not allowed.
- The type of each value must be compatible with the type of *expression*.
- The break statement is optional. If you omit the break, execution will continue on into the next case.

Switch Statement

```
switch (expression) {
case value1: // statement sequence
break;
case value2: // statement sequence
break;
.
.
.
default:
// default statement sequence
}
```

```
switch(a){
case 0:{
	result="Zero";
	break;
}
case 50:{
	result="Fifty";
	break;
}
case 100:{
	result="Hundread";
	break;
}
default:{
	result=Integer.toString(a);
	break;
}
}
```

Switch Statement

```
switch(count) {
case 1:
switch(target) { // nested switch
case 0:
System.out.println("target is zero");
break;
case 1: // no conflicts with outer switch
System.out.println("target is one");
break;
}
break;
case 2: // ...
```

```
switch (a) {
case 0: {
        result = "Zero";break;
}
case 50: {
        result = "Fifty";break;
}
case 100: {
        switch (a) {
        case 150: {
                result = "One Fifty";break;
        }
        case 200: {
                result = "Double Hundread";break;
        }
        }
        result = "Hundread";break;
}
default: {
        result = Integer.toString(a);break;
}
}
```

Source Code

Variable Naming convention

Questions?