Lecture 06 – Variables, Data Types and Control Statements

Variables, Identifiers and Data Types

Variables are used for data that can be modified during program execution. All variables have a name, a type, and a scope. The programmer assigns some names to variables, known as **identifiers**. An Identifier must be unique within a scope of the Java program. Variables have a **data type** that indicates the kind of value they can store.

Data Types

The ***data type*** indicates the attributes of the variable, such as the range of values that can be stored and the operators that can be used to manipulate the variable. Java has four main primitive data types built into the language. We can also create our own composite data types.

Java has four main primitive data types built into the language. We can also create our own data types.

- ***Integer:*** byte, short, int, and long
- ***Floating-point:*** float and double
- ***Character:*** char
- ***Boolean:*** true or false

The following chart summarizes the default values for the Java built-in data types.

Table 1: Java built-in data types

Data Type	**Default Value (for fields)**	**Range**
byte	0	-127 to +128
short	0	-32768 to +32767
int	0	-2,147,483,648 to 2,147,483,647
long	0L	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	0.0f	According to single-precision 32-bit IEEE 754 floating point format
double	0.0d	According to double-precision 64-bit IEEE 754 floating point format
char	'\u0000′	0 to 65535
boolean	false	Is not precisely defined

Variable Declaration

When we declare a variable we assign it an identifier and a data type.

Syntax

```
type variable = value;
```

The term "type" can be a primitive type or reference type (i.e. non-primitive type). Examples of reference types are String or any user-defined type.

For Example,

```
String msg = "hello world";
```

In the above statement, String is the *data type* for the identifier named **msg**. If we don't specify a value when the variable is declared, it will be assigned with the default value for its data type.

Identifier Naming Rules

- It can consist of upper and lower case letters, digits, dollar sign ($) and the underscore (_) character.
- It must begin with a letter, dollar sign, or an underscore.
- It is case sensitive.
- Keywords cannot be used as identifiers
- Within a given section of our program or scope, each user defined item must have a unique identifier.
- It can be of any length.

Literals

We may have noticed that the new keyword isn't used when initializing a variable of a primitive type. Primitive types are special data types built into the language; they are not objects created from a class. A *literal* is the source code representation of a fixed value; literals are represented directly in our code without requiring computation. As shown below, it's possible to assign a literal to a variable of a primitive type:

```
boolean result = true;
char ch = 'C';
byte b = 100;
short s = 10000;
int i = 100000;
```

Decision Control Statements

In Java programming, the decision control statements perform different actions depending on whether a specific condition holds true or false. Based on this, certain decisions are to be made. They are also called *conditional statements*.

We have the following decision control statements in Java:

- **if statement** – we use this statement if we want to execute some code only if a specified condition is true.

  Syntax:

  ```
  if (condition)
  {
     code to be executed if condition is true
  }
  ```

- **if-else statement** – we use this statement if we want to execute some code if the condition is true and another code if the condition is false.

  Syntax:

  ```
  if (condition)
  {
     code to be executed if condition is true
  }
  else
  {
     code to be executed if condition is not true
  }
  ```

- **if-else nesting statement** – we use this statement if we want to select one of many blocks of code to be executed.

  Syntax:

  ```
  if (condition1)
  {
    code to be executed if condition1 is true

  }
  else if (condition2)
  {
    code to be executed if condition2 is true

  }
  else
  {
    code to be executed if condition1 and condition2 are not true
  }
  ```

- **switch-case statement** – we use this statement if we want to select one of many blocks of code to be executed

  Syntax:

  ```
  switch(choice)
  {

      case 1:
         execute code block 1
         break;

      case 2:
         execute code block 2
         break;

      case 3:
         execute code block 3
         break;

      default:
         execute code if choice is different
         break;

  }
  ```

Loop Control Statements

Java loops are used to execute the same block of code a specified number of times or while a specified condition is true.

Sometimes, when we write code, we want the same block of code to run repeatedly in a program.

In Java there are mainly three different kinds of loops:

- **while loop** – loops through a block of code while a specified condition is true.

  Syntax:

  ```
  while (condition)
  {
     //code to be executed
  }
  ```

- **do-while loop –** a variant of the while loop which will always be executed at least once, even if the condition is false, because the code is executed before the condition is tested.

Syntax:

```
do
{
   //code to be executed

} while (condition);
```

- **for loop** – loops through a block of code a specified number of times

```
for (initialization part; condition part; re-initialization part)
{
    //code to be executed
}
```

There are two special statements that can be used inside loops: *break* and *continue*.

Besides this, there are two other variations of ***for loop*** — ***labeled for loop*** and ***enhanced for loop***.