Agenda

- Data Types
- Decision Making & Loops

Data

Data Types

Primitive Data Types

The Java programming language defines eight primitive types:

- Logical – boolean
- Textual – char
- Integral – byte, short, int, and long
- Floating – double and float

Logical –boolean

The boolean primitive has the following characteristics:

- The boolean data type has two literals, true andfalse.
- For example, the statement:

boolean truth = true;

declares the variable truth as boolean type and assigns it a value of true.

Textual –char

The textual char primitive has the following characteristics:

- Represents a 16-bit Unicode character
- Must have its literal enclosed in single quotes (’ ’)
- Uses the following notations:

'a'	The letter a
'\t'	The tab character
'\u????‘	A specific Unicode character, ????, is replaced with exactly four hexadecimal digits .

For example, ’\u03A6’ is the Greek letter phi [Φ].

Integral –byte,short,int, andlong

The integral primitives have the following characteristics:

- Integral primates use three forms: Decimal, octal, or hexadecimal
- Literals have a default type of int.
- Literals with the suffix L or l are of type long.

Integer Length	Name or Type	Range
8 bits	byte	-2^7 to 2^7-1
16 bits	short	-2^{15} to $2^{15}-1$
32 bits	int	-2^{31} to $2^{31}-1$
64 bits	long	-2^{63} to $2^{63}-1$

Floating Point –float and double

The floating point primitives have the following characteristics:

- Floating-point literal includes either a decimal point or one of the following:
- E or e (add exponential value)
- F or f (float)
- D or d (double)

Examples:

3.14	A simple floating-point value (a double)
6.02E23	A large floating-point value
2.718F	A simple float size value
123.4E+306D	A large double value with redundant D

Float Length	Name or Type
32 bits	`float`
64 bits	`double`

Primitive Data Types contd..

- byte:
  - Byte data type is a 8-bit signed two's complement integer.
  - Minimum value is -128 (-2^7)
  - Maximum value is 127 (inclusive)(2^7 -1)
  - Default value is 0
  - Byte data type is used to save space in large arrays, mainly in place of integers, since a byte is four times smaller than an int.
  - Example : byte a = 100 , byte b = -50

- short:
  - Short data type is a 16-bit signed two's complement integer.
  - Minimum value is -32,768 (-2^15)
  - Maximum value is 32,767(inclusive) (2^15 -1)
  - Short data type can also be used to save memory as byte data type. A short is 2 times smaller than an int
  - Default value is 0.
  - Example : short s= 10000 , short r = -20000

- int:
  - Int data type is a 32-bit signed two's complement integer.
  - Minimum value is - 2,147,483,648.(-2^31)
  - Maximum value is 2,147,483,647(inclusive).(2^31 -1)
  - Int is generally used as the default data type for integral values unless there is a concern about memory.
  - The default value is 0.
  - Example : int a = 100000, int b = -200000

- long:
  - Long data type is a 64-bit signed two's complement integer.
  - Minimum value is -9,223,372,036,854,775,808.(-2^63)
  - Maximum value is 9,223,372,036,854,775,807 (inclusive). (2^63 -1)
  - This type is used when a wider range than int is needed.
  - Default value is 0L.
  - Example : long a = 100000L, int b = -200000L

- float:
  - Float data type is a single-precision 32-bit IEEE 754 floating point.
  - Float is mainly used to save memory in large arrays of floating point numbers.
  - Default value is 0.0f.
  - Float data type is never used for precise values such as currency.
  - Example : float f1 = 234.5f

- double:
  - double data type is a double-precision 64-bit IEEE 754 floating point.
  - This data type is generally used as the default data type for decimal values. generally the default choice.
  - Default value is 0.0d.
  - Example : double d1 = 123.4

- boolean:
  - boolean data type represents one bit of information.
  - There are only two possible values : true and false.
  - This data type is used for simple flags that track true/false conditions.
  - Default value is false.
  - Example : boolean one = true

- char:
  - char data type is a single 16-bit Unicode character.
  - Minimum value is '\u0000' (or 0).
  - Maximum value is '\uffff' (or 65,535 inclusive).
  - Char data type is used to store any character.
  - Example . char letterA ='A'

Reference Data Types

- Reference variables are created using defined constructors of the classes.
- Class objects, and various type of array variables come under reference data type.
- Default value of any reference variable is null.
- A reference variable can be used to refer to any object of the declared type or any compatible type.
  - Example : Animal animal = new Animal("giraffe");

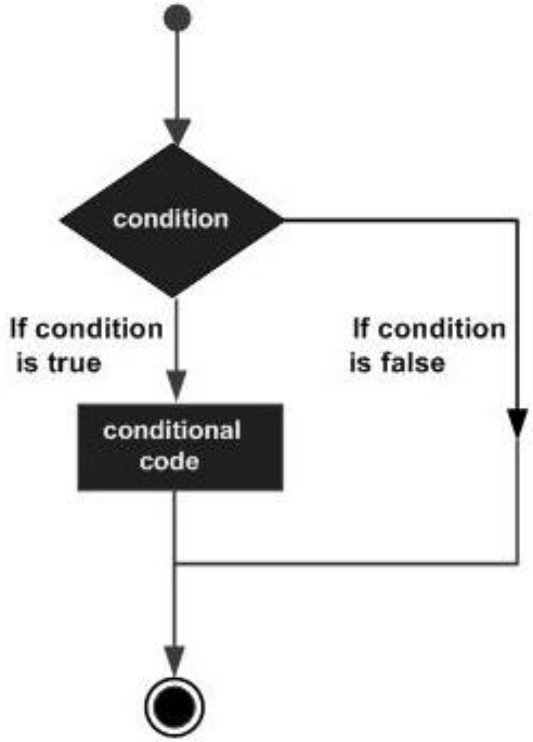
DECISION MAKING & LOOPS

Decisions and Loops

- A ***decision*** *lets you run either one section of code or another, based on the results of a specific test.*
    - Making decisions with the if , else , and switch statements
    - compact decision code with the ternary operator

- A ***loop*** *lets you run the same section of code over and over again until a specific* condition is met.
    - Looping with the do , while , and for statements
    - Altering loops with the break and continue statements
    - Nesting loops inside each other

Making Decisions



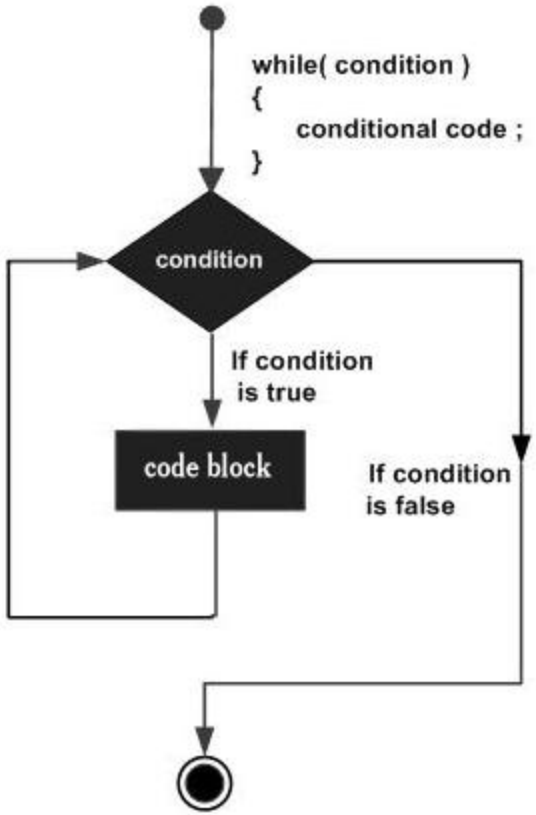
Loop Controls

- while Loop
- do...while Loop
- for Loop
- Break
- continue

while

- while(Boolean_expression) { //Statements }

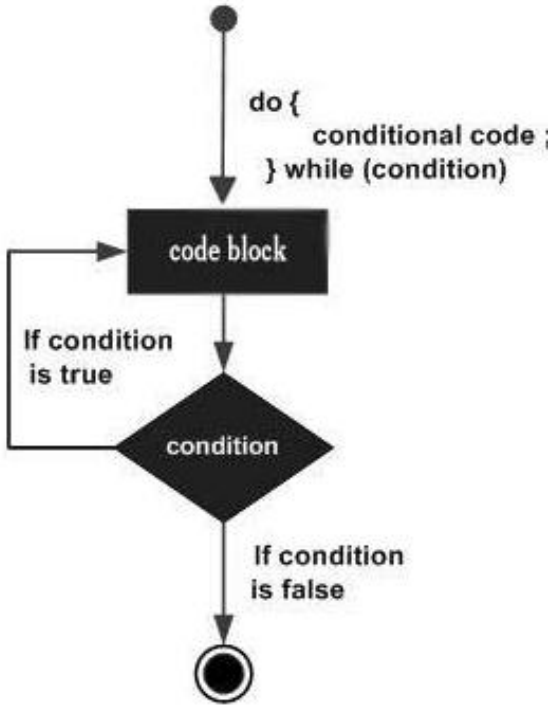
```
public class Test {

   public static void main(String args[]) {
      int x = 10;

      while( x < 20 ) {
         System.out.print("value of x : " + x
);
         x++;
         System.out.print("\n");
      }
   }
}
```



Do..while

- do { //Statements }while(Boolean_expression);

```
public class Test {

   public static void main(String args[]){
      int x = 10;

      do{
         System.out.print("value of x : " + x );
         x++;
         System.out.print("\n");
      }while( x < 20 );
   }
}
```

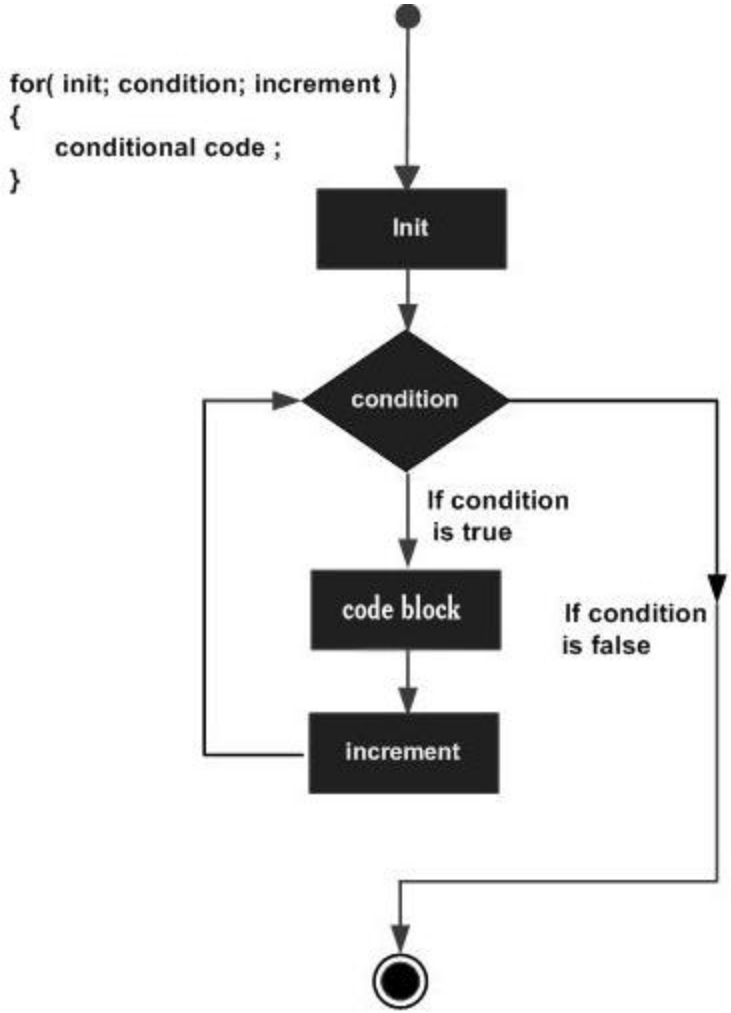


For loop

for(initialization; Boolean_expression; update)

{ //Statements }

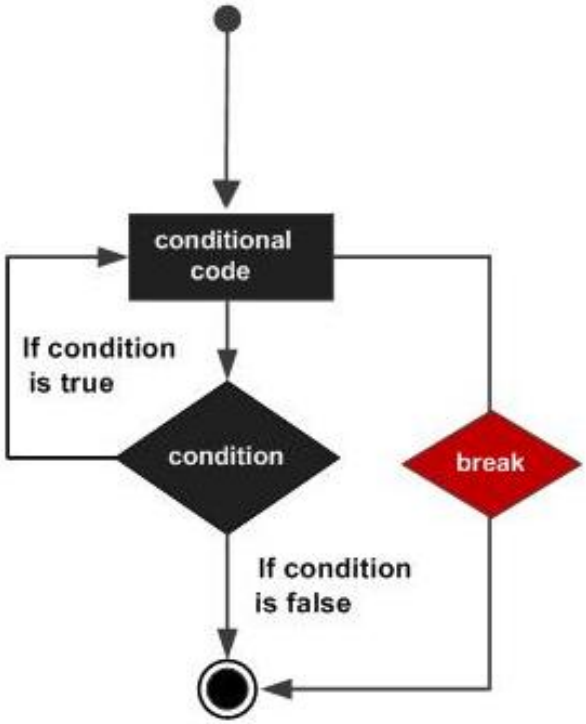
```
public class Test {

   public static void main(String args[]) {

      for(int x = 10; x < 20; x = x+1)
  {
     System.out.print("value of x : " + x );
     System.out.print("\n");
  }
}
}
```



The break Keyword

- to stop the entire loop
- must be used inside any loop or a switch statement

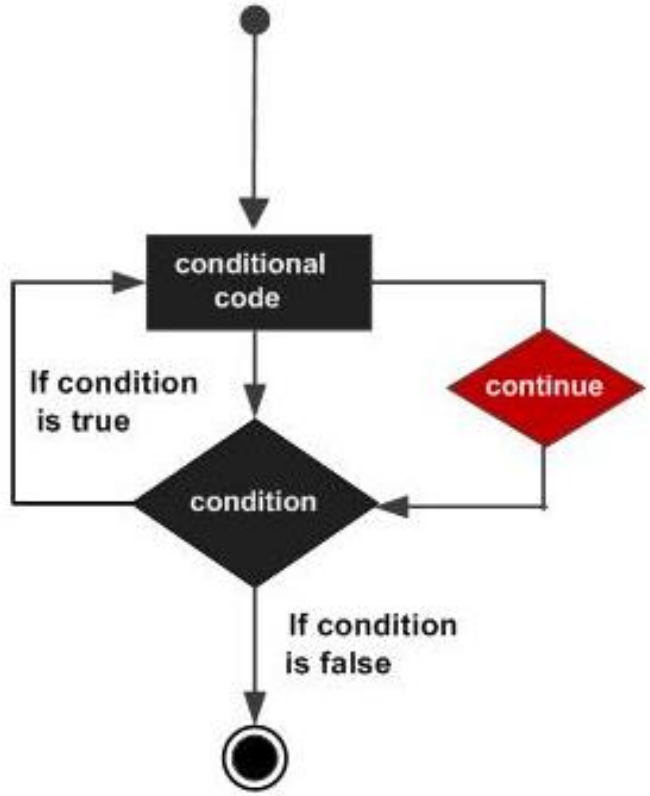
```
public class Test {

   public static void main(String args[]) {
      int [] numbers = {10, 20, 30, 40, 50};

      for(int x : numbers ) {
         if( x == 30 ) {
                break;
         }
         System.out.print( x );
         System.out.print("\n");
      }
   }
}
```



The continue Keyword

- used in any of the loop control structures.
- causes the loop to immediately jump to the next iteration of the loop
- In a for loop, the continue keyword causes flow of control to immediately jump to the update statement.
- In a while loop or do/while loop, flow of control immediately jumps to the Boolean expression.

```
public class Test {

   public static void main(String args[]) {
      int [] numbers = {10, 20, 30, 40, 50};

      for(int x : numbers ) {
         if( x == 30 ) {
                continue;
         }
         System.out.print( x );
         System.out.print("\n");
      }
   }
}
```



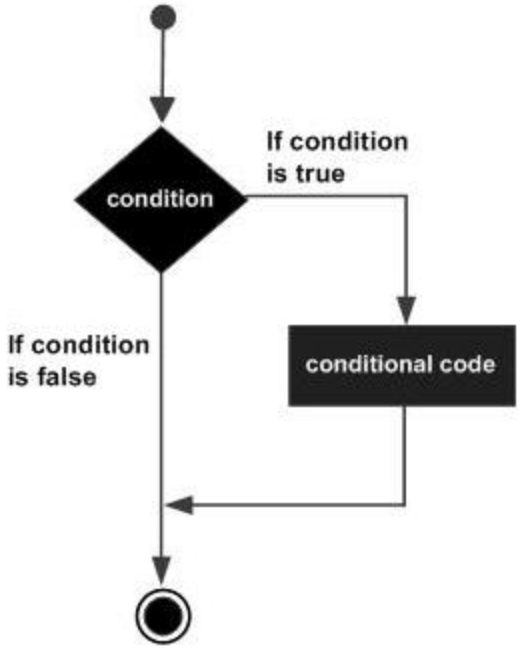
Decision Making

- If / if..else
- Switch

The if Statement

- An if statement consists of a Boolean expression followed by one or more statements.
- if(Boolean_expression) { //Statements will execute if the Boolean expression is true }

```
public class Test {

   public static void main(String args[])
      {
       int x = 10;
       if( x < 20 )
       {
          System.out.print("This is if
statement");
       }
    }
}
```



The if...else Statement

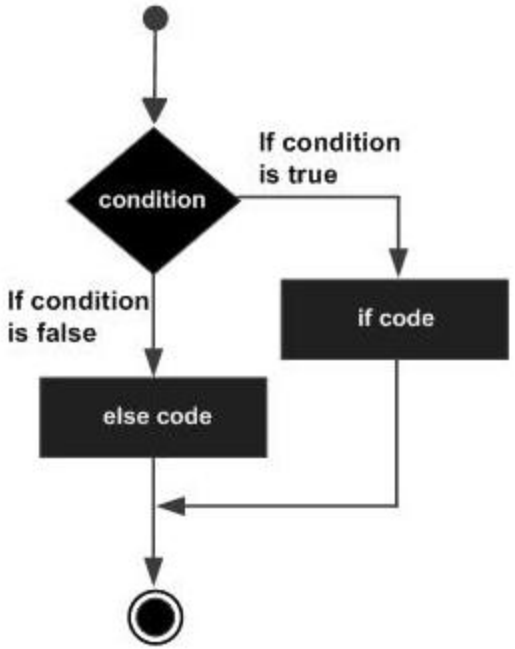
if(Boolean_expression)

{ //Executes when the Boolean expression is true }

Else

{ //Executes when the Boolean expression is false }

```
public class Test {

   public static void main(String args[]){
      int x = 30;

      if( x < 20 ){
         System.out.print("This is if statement");
      }else{
         System.out.print("This is else statement");
      }
   }
}
```



The if...else if...else Statement

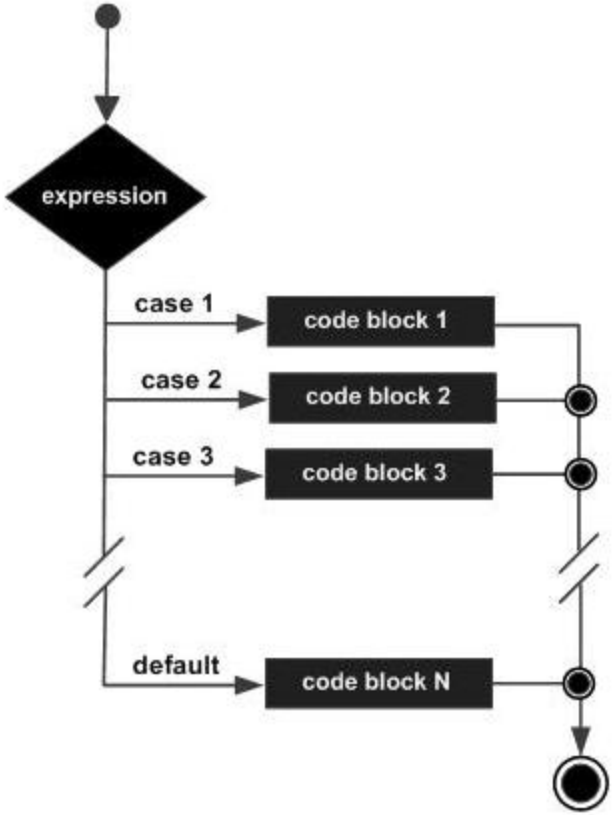
```
if(Boolean_expression 1)
{ //Executes when the Boolean expression 1 is true }
else if(Boolean_expression 2)
{ //Executes when the Boolean expression 2 is true }
else if(Boolean_expression 3)
{ //Executes when the Boolean expression 3 is true }
else
{ //Executes when the none of the above condition is true. }
```

```
public class Test {

   public static void main(String args[]){
      int x = 30;

      if( x == 10 ){
         System.out.print("Value of X is 10");
      }else if( x == 20 ){
         System.out.print("Value of X is 20");
      }else if( x == 30 ){
         System.out.print("Value of X is 30");
      }else{
         System.out.print("This is else statement");
      }
   }
}
```

The switch Statement

- A *switch* statement allows a variable to be tested for equality against a list of values.
- Each value is called a case, and the variable being switched on is checked for each case.

```
switch(expression)
{ case value :
//Statements
break;
 //optional
case value :
//Statements
break;
//optional
//You can have any number of case statements.
default :
 //Optional
//Statements
}
```



```
public class Test {

   public static void main(String args[]){
      //char grade = args[0].charAt(0);
      char grade = 'C';

      switch(grade)
      {
         case 'A' :
            System.out.println("Excellent!");
            break;
         case 'B' :
         case 'C' :
            System.out.println("Well done");
            break;
         case 'D' :
            System.out.println("You passed");
         case 'F' :
            System.out.println("Better try again");
            break;
         default :
            System.out.println("Invalid grade");
      }
      System.out.println("Your grade is " + grade);
   }
}
```

Thank You