Java Basic Programming Constructs

```
/*
 * This is your first java program.
 */

class HelloWorld{
  public static void main(String[] args){
    System.out.println("Hello World!");
  }
}
```

A Closer Look at HelloWorld

This is a comment!

```
/*
 * This is your first java program.
 */

class HelloWorld{
  public static void main(String[] args){
    System.out.println("Hello World!");
  }
}
```

A Closer Look at HelloWorld

This is a class named “HelloWorld”.

```
/*
 * This is your first java program.
 */

class HelloWorld{
  public static void main(String[] args){
    System.out.println(“Hello World!”);
  }
}
```

A Closer Look at HelloWorld

The main function of your program.

```
/*
 * This is your first java program.
 */

class HelloWorld{
  public static void main(String[] args){
    System.out.println("Hello World!");
  }
}
```

A Closer Look at HelloWorld

The arguments of the program.

```
/*
 * This is your first java program.
 */

class HelloWorld{
  public static void main(String[] args){
    System.out.println("Hello World!");
  }
}
```

A Closer Look at HelloWorld

Output statement of your program.

```
/*
 * This is your first java program.
 */

class HelloWorld{
  public static void main(String[] args){
    System.out.println("Hello World!");
  }
}
```

A Closer Look at HelloWorld

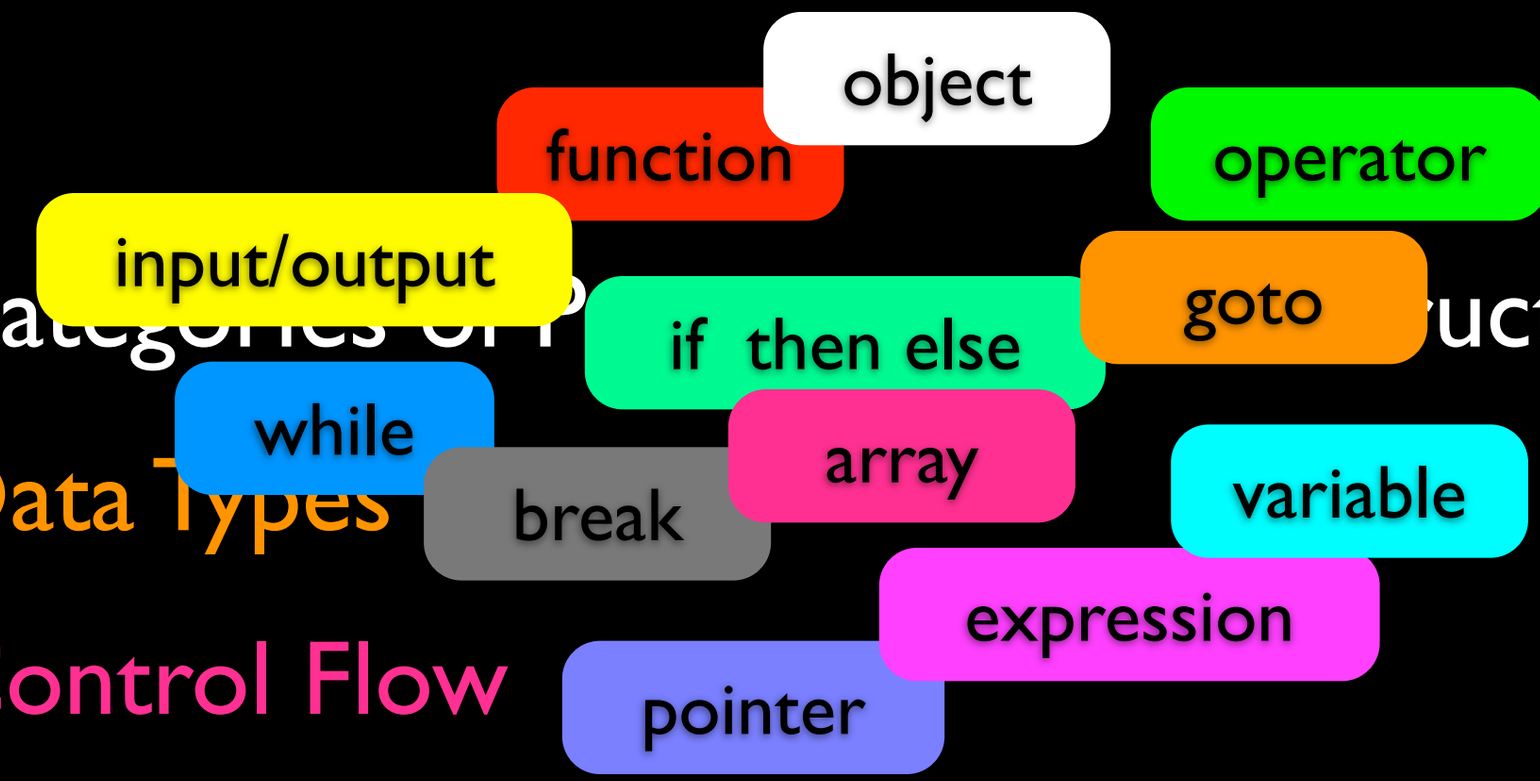
```
/*
 * This is your first java program.
 */

class HelloWorld{
  public static void main(String[] args){
    System.out.println("Hello World!");
  }
}
```

A Closer Look at HelloWorld

Programming Constructs

Programming Constructs

- 2 Categories of Programming Constructs
  - Data Types
  - Control Flow



```java
/*
 * This is your second java program.
 */

class AnotherExample{
  public static void main(String[] args){
    int i = 20;
    System.out.println("The i is : " + i);
    i = i * i;
    System.out.print("The i ^ 2 is : ");
    System.out.println(i);
  }
}
```

Another Example

```
/*
 * This is your second java program.
 */
class AnotherExample{
  public static void main(String[] args){
    int i = 20;
    System.out.println("The i is : " + i);
    i = i * i;
    System.out.print("The i ^ 2 is : ");
    System.out.println(i);
  }
}
```

Another Closer Look!

- A variable is defined as integer.
- Its name is “i”.
- It is initialized to 20.

```
class AnotherExample{
  public static void main(String[] args){
    int i = 20;
    System.out.println("The i is : " + i);
    i = i * i;
    System.out.print("The i ^ 2 is : ");
    System.out.println(i);
  }
}
```

Another Closer Look!

- This is a string literal.
- It contains “The i is : ”
- Its length is 11.

```
class AnotherExample{
  public static void main(String[] args){
    int i = 20;
    System.out.println("The i is : " + i);
    i = i * i;
    System.out.print("The i ^ 2 is : ");
    System.out.println(i);
  }
}
```

Another Closer Look!

- “+” concatenates strings in Java.
- It can also concatenate integer with a string.
- The result is a string.

```
class AnotherExample{
  public static void main(String[] args){
    int i = 20;
    System.out.println("The i is : " + i);
    i = i * i;
    System.out.print("The i ^ 2 is : ");
    System.out.println(i);
  }
}
```

Another Closer Look!

- The print functions writes without newline.
- The println functions writes with newline.

```
class AnotherExample{
  public static void main(String[] args){
    int i = 20;
    System.out.println("The i is : " + i);
    i = i * i;
    System.out.print("The i ^ 2 is : ");
    System.out.println(i);
  }
}
```

Another Closer Look!

```
class AnotherExample{
  public static void main(String[] args){
    int i = 20;
    System.out.println("The i is : " + i);
    i = i * i;
    System.out.print("The i ^ 2 is : ");
    System.out.println(i);
  }
}
```

Another Closer Look!

```
/*
 * This is your third java program.
 */

class YetAnotherExample{
  public static void main(String[] args){
    int i = 20;
    if ( i<30 )
      System.out.println(" i < 30 “);
    else
      System.out.println(“ i > 30”);
  }
}
```

Yet Another Example

Yet Another Example

- If statement!
- Else statement!

```
class YetAnotherExample{
  public static void main(String[] args){
    int i = 20;
    if ( i<30 )
      System.out.println(" i < 30 “);
    else
        System.out.println(“ i > 30”);
  }
}
```

- A boolean expression.
- It is true now.

```
class YetAnotherExample{
  public static void main(String[] args){
    int i = 20;
    if ( i<30 )
      System.out.println(" i < 30 “);
    else
      System.out.println(“ i > 30”);
  }
}
```

Yet Another Example

```
class YetAnotherExample{
  public static void main(String[] args){
    int i = 20;
    if ( i<30 )
      System.out.println(" i < 30 “);
    else
      System.out.println(“ i > 30”);
  }
}
```

Yet Another Example

Java Programming Constructs

Blocks

```
{
  ...
}


if(x < 30){
  y = 20;
  x = 30 * 30;
}
```

Blocks

- Blocks are defined with braces like C.
- Why blocks are used ?
  - Sometimes a group of statements needed to be executed in all or nothing manner.
- A block is the same as a single statement.

```
{
   ...
}


if(x < 30){
  y = 20;
  x = 30 * 30;
}
```

Variables

```
int i;
double i, j;
long i = 10, j;
```

Variables

- 2 types of variable
  - Class variable (Fields)
  - Local variables
- There is no global variable.
- Variables can be defined in any place inside a class.
- Variables can be defined and initialized.
- Java uses static bindings.
- A variable always refers to its nearest enclosing binding.

```
int i;
double i, j;
long i = 10, j;
```

Operators

```
i = i - j;
i = i * i;
s = “aaa” + i;
i++;
i -= 1;
```

Operators

- An operator in Java is the same as operators in C++, but they cannot be overrode.

- =, + , -, ==, =, -=, ...

```
i = i - j;
i = i * i;
s = “aaa” + i;
i++;
i -= 1;
```

Constants

```
1, 1d, 1l, 1.1, 0xFF, 077

‘c’, ‘A’

“”, “Java”, “cA”

true, false

null
```

Constants

- Numerical constants `1, 1d, 1l, 1.1, 0xFF, 077`
- Characters `‘c’, ‘A’`
- String literals `“”, “Java”, “cA”`
- Booleans `true, false`
- nil `null`

Expression

```
j == i, !i

j = !(i == k), j = i = 1
```

Expression

- Boolean Expression `j == i, !i`
- Assignment Expression `j = !(i == k), j = i = 1`
- ....

Statement

```
i = 1;

{ ... }

if(...) { } else { }

while(...) { }

for(...) { }
```

Statement

- Expression + “;” `i = 1;`
- Block `{ ... }`
- If Statement `if(...) { } else { }`
- While Statement `while(...) { }`
- For Statement `for(...) { }`

Primitive Data Types

Primitive Data Types

- Integers
  - byte, short, int, long.
- Floating points
  - float, double.
- Characters
  - char
- Boolean
  - boolean
- void

Arrays

```
<TYPE> name[][]...[] = new
<TYPE>[SIZE][SIZE]...[SIZE];

int oneDimArr[] =
  new int[20];

int threeDimArr[][][] =
  new int[10][20][30];

int[] oneDimArr2 =
  new int[20];

int sizeOfArr =
  threeDimArr.length;
```

Arrays

- Arrays are very similar to C.
- The main difference is in memory allocation.
- There are two syntax for array definition.
- The length property contains the size of your array.

```
<TYPE> name[][]...[] = new
<TYPE>[SIZE][SIZE]...[SIZE];

int oneDimArr[] =
  new int[20];

int threeDimArr[][][] =
  new int[10][20][30];

int[] oneDimArr2 =
  new int[20];

int sizeOfArr =
  threeDimArr.length;
```

String

```
String str = “abcd”;

str = str + “abcde”;

str = 1 + “abcd”;
```

String

- String is a type in Java.
- String literals are similar to C.
- We have talked about the “+” operator.
  - It can concatenate anything to String.

```
String str = “abcd”;

str = str + “abcde”;

str = 1 + “abcd”;
```

Arithmatic Operators

Arithmatic Operators

- + : Add
- - : Subtract
- / : Divide
- * : Multiply
- % : Modulus
- ++ : Increment
- -- : Decrement
- += : Addition Assignment
- -= : Subtraction Asg.
- *= : Multiplication Asg.
- /= : Division Asg.
- %= : Modulus Asg.

Bitwise Operators

Bitwise Operators

- ~ : NOT
- & : AND
- | : OR
- ^ : XOR
- >> : Shift Right
- >>> : Shift Right Zero Fill
- << : Shift Left
- &= : AND Assignment
- |= : OR Assignment
- ^= : XOR Assigment

Boolean Operators

Boolean Operators

- & : Logical AND
- | : Logical OR
- ^ : Logical XOR
- || : Short circuit OR
- && : Short circuit AND
- ! : Logical NOT
- &= : AND Assignment
- |= : OR Assignment
- ^= : XOR Assignment
- == : XOR Assignment
- != : Not equal to
- ? : : Ternary if-then-else

Warning

```
(x != 0) & (1/x != 2)
```

```
(x != 0) && (1/x != 2)
```

Warning

- If X is Zero, you will get DivisionByZero error from `(x != 0) & (1/x != 2)`
- But, you won't get the error from `(x != 0) && (1/x != 2)`

Control Statements

Control Statements

- If statement
- While statement
- For statement
- Do-While statement
- Switch statement
- return, break, continue

if ... else ...

```
if ( x == 1 )
  System.out.println(“x was 1”);
else {
  System.out.println(“x was not 1”);
}
```

if ... else ...

- if (boolean-expr) stmt else stmt

```
if ( x == 1 )
  System.out.println(“x was 1”);
else {
  System.out.println(“x was not 1”);
}
```

while ...

```
while(x > 0)
  System.out.println(“x is ” + x--);
```

while ...

- while (boolean-expr) stmt

```
while(x > 0)
  System.out.println(“x is ” + x--);
```

for ...

```
for (int i = 0; i < 100; i++)
  System.out.println(“i is “ + i);
```

for ...

- for(inital-stmt; conditions; stmts) stmt

```
for (int i = 0; i < 100; i++)
  System.out.println(“i is “ + i);
```

do ... while ...

```
do
  System.out.println(“i is “ + i);
while (i-- > 0);
```

do ... while ...

- do stmt while(boolean-expr);

```
do
  System.out.println(“i is “ + i);
while (i-- > 0);
```

switch ... case ...

```
switch ( i ) {
  case 1:
    System.out.println(“one”);
    break;
  case 2:
    System.out.println(“two”);
    break;
  default:
    System.out.println(“I cannot count more than two”);
}
```

switch ... case ...

- switch(expr) { exp1: stmt; exp2: stmt ... }

```
switch ( i ) {
  case 1:
    System.out.println(“one”);
    break;
  case 2:
    System.out.println(“two”);
    break;
  default:
    System.out.println(“I cannot count more than two”);
}
```

return, break, continue

```
return 1;
```

```
for (;;) {
  continue;
}
```

```
for (;;) {
  break;
}
```

return, break, continue

- return

```
return 1;
```

  - Sets the return value of a function and exit from it.

- break

  - Jumps out of a block.

```
for (;;) {
  break;
}
```

- continue

  - Jumps to the beginning of a block.

```
for (;;) {
  continue;
}
```

Goto in Java

```
first: {
  second: {
    third: {
      System.out.println(“B
      efore 3rd”);
      break second;
      System.out.println(“3
      rd”);
    }
    System.out.println(“sec
    ond”);
  }
  System.out.println(“firs
  t”);
}
```

Goto in Java

- break <label>;
- Break can be used to jump out of a block with the label.

```
first: {
  second: {
    third: {
      System.out.println("B
      efore 3rd");
      break second;
      System.out.println("3
      rd");
    }
    System.out.println("sec
    ond");
  }
  System.out.println("firs
  t");
}
```

Some Useful Functions

Some Useful Functions

- Integer.parseInt() converts a string to an integer.
  - Integer.parseInt("1") == 1 is true
- Double.parseDouble() converts a string to a double.
  - Double.parseDouble("1.1") == 1.1 is true.

Any Questions?