Java Extension Packages

```java
import <class>
```

Packages

```java
package package_path package_name;
```

Common Extensions

```java
java.awt, java.io, java.lang, java.util, javax.swing
```

Data Types

```java
boolean, char, byte, short, int , long, float, double, String
```

Comments

```java
// Single line Comment
/* Multiple line Comment */
```

Arithmetic Operators

```java
+ (Addition), - (Subtraction), * (Multiplication), / (Division), % (Modulus)
```

Equality Operators

```java
== (Equal To), != (Not Equal To)
```

Relational Operators

```java
> (Greater Than), < (Less Than), >= (Greater Than or Equal To), <= (Less Than or Equal To)
```

Increment/Decrement Operators

```java
++x (PreIncrement), x++ (PostIncrement), --x (PreDecrement), x-- (PostDecrement)
```

Logical Operators

```java
&& (boolean AND), & (boolean logical AND), || (logical OR), | (boolean logical inclusive OR), ^ (boolean logical exclusive OR), ! (logical NOT)
```

Escape Sequences

```java
\n (newline), \t (horizontal tab), \v (carriage return), \b (backslash), \" (double quote)
```

Other

```java
? (Conditional), = (Assignment)
```

If Else

```java
if (condition) {
  <statement(s>);
} else{
  <statement(s>);
}
```

Switch Case

```java
switch(expression){
  case <option 1>:
    <statement>
    break;
  case <option 2>:
    <statement>
    break;
[default:
  <statement>;
  ]
}
```

For Loop

```java
for (<initial value>; <condition>; <in-/decrement>){
  <statement(s>);
}
```

While Loop

```java
while (<condition> )
{
  <statement(s)>;
}
```

Do While Loop

```java
do {
  <statement(s)>;
} while (<condition>);
```

Arrays

```java
int c[] = new int[5]; //declare and allocate in one
//declare and allocate in two
int myArray[];
myArray = new int[5];
//initialize
myArray = {10,20,30,40,50}
//access 3rd Element
myArray[2] = var;
```

Method

```java
<access modifier> <return data type> <function name> <parameters>
{
  <declarations>
  <statements>
  [return;]
  [return <expression>;]
}
```

Class

```java
<access modifier> <return data type> <class name> [extends <superclass name>] [implements <interface name>] 
{
  <declarations>
  <methods>
}
```

Exception Handling

```java
try{
  //Code, can include method calls
} catch(Exception e){
  //What to do on error. Multiple catches may be used
} finally{
  //this code is executed with or without an error
}
```

File IO

```java
// Read in a Text File
//should be contained in a try catch block
BufferedReader in = new BufferedReader(new FileReader(directory.getPath()));
//directory is a File object
String nextLine = in.readLine(); //reads first line, repeat for next line
in.close();

// Write to a Text File
//should be contained in a try catch block
DataOutputStream out = new DataOutputStream(new FileOutputStream(myfile.dat));
//creates myfile dat, can add directory
out.writeUTF(\"the text\");
```

Download More Reference Sheets & Get Programming Help @
http://www.dreamincode.net

Edited By: SPuzzle, Jiri, Willian, Wilkens, Ichir, Peter Vogel, JaveniBK
Published: August 25, 2008