Java	Program Structure	C#

```java
package hello;

public class HelloWorld {
  public static void main(String[] args) {
    String name = "Java";

    // See if an argument was passed from the command line
    if (args.length == 1)
      name = args[0];

    System.out.println("Hello, " + name + "!");
  }
}
```

```csharp
using System;

namespace Hello {
  public class HelloWorld {
    public static void Main(string[] args) {
      string name = "C#";

      // See if an argument was passed from the command line
      if (args.Length == 1)
        name = args[0];

      Console.WriteLine("Hello, " + name + "!");
    }
  }
}
```

Java	Comments	C#

```java
// Single line
/* Multiple
   line  */
/** Javadoc documentation comments */
```

```csharp
// Single line
/* Multiple
   line  */
/// XML comments on a single line
/** XML comments on multiple lines */
```

Java	Data Types	C#

Primitive Types
boolean
byte
char

Value Types
bool
byte, sbyte
char

short, int, long
float, double

Reference Types
Object *(superclass of all other classes)*
String
arrays, classes, interfaces

Conversions

```
// int to String
int x = 123;
String y = Integer.toString(x);  // y is "123"

// String to int
y = "456";
x = Integer.parseInt(y);   // x is 456

// double to int
double z = 3.5;
x = (int) z;   // x is 3  (truncates decimal)
```

short, ushort, int, uint, long, ulong
float, double, decimal
structures, enumerations

Reference Types
object *(superclass of all other classes)*
string
arrays, classes, interfaces, delegates

Convertions

```
// int to string
int x = 123;
String y = x.ToString();  // y is "123"

// string to int
y = "456";
x = int.Parse(y);   // or x = Convert.ToInt32(y);

// double to int
double z = 3.5;
x = (int) z;   // x is 3  (truncates decimal)
```

Java	Constants	C#

```
// May be initialized in a constructor
final double PI = 3.14;
```

```
const double PI = 3.14;

// Can be set to a const or a variable. May be initialized in a constructor.
readonly int MAX_HEIGHT = 9;
```

Java	Enumerations	C#

```
enum Action {Start, Stop, Rewind, Forward};

// Special type of class
enum Status {
  Flunk(50), Pass(70), Excel(90);
  private final int value;
  Status(int value) { this.value = value; }
  public int value() { return value; }
};

Action a = Action.Stop;
if (a != Action.Start)
  System.out.println(a);          // Prints "Stop"

Status s = Status.Pass;
System.out.println(s.value());    // Prints "70"
```

```
enum Action {Start, Stop, Rewind, Forward};

enum Status {Flunk = 50, Pass = 70, Excel = 90};

No equivalent.

Action a = Action.Stop;
if (a != Action.Start)
  Console.WriteLine(a);           // Prints "Stop"

Status s = Status.Pass;
Console.WriteLine((int) s);       // Prints "70"
```

Java Operators C#

```
Comparison
== < > <= >= !=

Arithmetic
+ - * /
%  (mod)
/  (integer division if both operands are ints)
Math.Pow(x, y)

Assignment
= += -= *= /= %= &= |= ^= <<= >>= >>>= ++ --

Bitwise
& | ^ ~ << >> >>>
```

```
Comparison
== < > <= >= !=

Arithmetic
+ - * /
%  (mod)
/  (integer division if both operands are ints)
Math.Pow(x, y)

Assignment
= += -= *= /= %= &= |= ^= <<= >>= ++ --

Bitwise
& | ^ ~ << >>
```

Logical
&& || & | ^ !

Note: && and || perform short-circuit logical evaluations

String Concatenation
+

Logical
&& || & | ^ !

Note: && and || perform short-circuit logical evaluations

String Concatenation
+

Java Choices C#

```
greeting = age < 20 ? "What's up?" : "Hello";

if (x < y)
  System.out.println("greater");

if (x != 100) {
  x *= 5;
  y *= 2;
}
else
  z *= 6;

int selection = 2;
switch (selection) {     // Must be byte, short, int, char, or enum
  case 1: x++;           // Falls through to next case if no break
  case 2: y++;   break;
  case 3: z++;   break;
  default: other++;
}
```

```
greeting = age < 20 ? "What's up?" : "Hello";

if (x < y)
  Console.WriteLine("greater");

if (x != 100) {
  x *= 5;
  y *= 2;
}
else
  z *= 6;

string color = "red";
switch (color) {                  // Can be any predefined type
  case "red":    r++;    break;   // break is mandatory; no fall-
through
  case "blue":   b++;    break;
  case "green": g++;     break;
  default: other++;      break;    // break necessary on default
}
```

Java Loops C#

```java
while (i < 10)
  i++;

for (i = 2; i <= 10; i += 2)
  System.out.println(i);

do
  i++;
while (i < 10);

for (int i : numArray)  // foreach construct
  sum += i;

// for loop can be used to iterate through any Collection
import java.util.ArrayList;
ArrayList<Object> list = new ArrayList<Object>();
list.add(10);    // boxing converts to instance of Integer
list.add("Bisons");
list.add(2.3);    // boxing converts to instance of Double

for (Object o : list)
  System.out.println(o);
```

```c#
while (i < 10)
  i++;

for (i = 2; i <= 10; i += 2)
  Console.WriteLine(i);

do
  i++;
while (i < 10);

foreach (int i in numArray)
  sum += i;

// foreach can be used to iterate through any collection
using System.Collections;
ArrayList list = new ArrayList();
list.Add(10);
list.Add("Bisons");
list.Add(2.3);

foreach (Object o in list)
  Console.WriteLine(o);
```

Java Arrays C#

```java
int nums[] = {1, 2, 3};   or   int[] nums = {1, 2, 3};
for (int i = 0; i < nums.length; i++)
  System.out.println(nums[i]);

String names[] = new String[5];
names[0] = "David";

float twoD[][] = new float[rows][cols];
twoD[2][0] = 4.5;
```

```c#
int[] nums = {1, 2, 3};
for (int i = 0; i < nums.Length; i++)
  Console.WriteLine(nums[i]);

string[] names = new string[5];
names[0] = "David";

float[,] twoD = new float[rows, cols];
twoD[2,0] = 4.5f;
```

```
int[][] jagged = new int[5][];
jagged[0] = new int[5];
jagged[1] = new int[2];
jagged[2] = new int[3];
jagged[0][4] = 5;
```

```
int[][] jagged = new int[3][] {
    new int[5], new int[2], new int[3] };
jagged[0][4] = 5;
```

Java Functions C#

```
// Return single value
int Add(int x, int y) {
  return x + y;
}

int sum = Add(2, 3);
```

```
// Return no value
void PrintSum(int x, int y) {
  System.out.println(x + y);
}

PrintSum(2, 3);
```

```
// Return single value
int Add(int x, int y) {
  return x + y;
}

int sum = Add(2, 3);
```

```
// Return no value
void PrintSum(int x, int y) {
  Console.WriteLine(x + y);
}

PrintSum(2, 3);
```

```
// Primitive types and references are always passed by value
void TestFunc(int x, Point p) {
  x++;
  p.x++;       // Modifying property of the object
  p = null;    // Remove local reference to object
}

class Point {
  public int x, y;
}

Point p = new Point();
p.x = 2;
int a = 1;
TestFunc(a, p);
System.out.println(a + " " + p.x + " " + (p == null) );  // 1 3 false
```

```
// Pass by value (default), in/out-reference (ref), and out-reference (out)
void TestFunc(int x, ref int y, out int z, Point p1, ref Point p2) {
  x++;  y++;  z = 5;
  p1.x++;       // Modifying property of the object
  p1 = null;    // Remove local reference to object
  p2 = null;    // Free the object
}

class Point {
  public int x, y;
}

Point p1 = new Point();
Point p2 = new Point();
p1.x = 2;
```

```
// Accept variable number of arguments
int Sum(int ... nums) {
  int sum = 0;
  for (int i : nums)
    sum += i;
  return sum;
}

int total = Sum(4, 3, 2, 1);   // returns 10
```

```
int a = 1, b = 1, c;   // Output param doesn't need initializing
TestFunc(a, ref b, out c, p1, ref p2);
Console.WriteLine("{0} {1} {2} {3} {4}",
   a, b, c, p1.x, p2 == null);   // 1 2 5 3 True

// Accept variable number of arguments
int Sum(params int[] nums) {
  int sum = 0;
  foreach (int i in nums)
    sum += i;
  return sum;
}

int total = Sum(4, 3, 2, 1);   // returns 10
```

Java Strings C#

```
// String concatenation
String school = "Harding ";
school = school + "University";   // school is "Harding University"

// String comparison
String mascot = "Bisons";
if (mascot == "Bisons")    // Not the correct way to do string comparisons
if (mascot.equals("Bisons"))   // true
if (mascot.equalsIgnoreCase("BISONS"))   // true
if (mascot.compareTo("Bisons") == 0)   // true

System.out.println(mascot.substring(2, 5));   // Prints "son"

// My birthday: Oct 12, 1973
java.util.Calendar c = new java.util.GregorianCalendar(1973, 10, 12);
String s = String.format("My birthday: %1$tb %1$te, %1$tY", c);
```

```
// String concatenation
string school = "Harding ";
school = school + "University";   // school is "Harding University"

// String comparison
string mascot = "Bisons";
if (mascot == "Bisons")    // true
if (mascot.Equals("Bisons"))   // true
if (mascot.ToUpper().Equals("BISONS"))   // true
if (mascot.CompareTo("Bisons") == 0)    // true

Console.WriteLine(mascot.Substring(2, 3));    // Prints "son"

// My birthday: Oct 12, 1973
DateTime dt = new DateTime(1973, 10, 12);
string s = "My birthday: " + dt.ToString("MMM dd, yyyy");
```

```
// Mutable string
StringBuffer buffer = new StringBuffer("two ");
buffer.append("three ");
buffer.insert(0, "one ");
buffer.replace(4, 7, "TWO");
System.out.println(buffer);     // Prints "one TWO three"
```

```
// Mutable string
System.Text.StringBuilder buffer = new
System.Text.StringBuilder("two ");
buffer.Append("three ");
buffer.Insert(0, "one ");
buffer.Replace("two", "TWO");
Console.WriteLine(buffer);     // Prints "one TWO three"
```

Java	Exception Handling	C#

```
// Must be in a method that is declared to throw this exception
Exception ex = new Exception("Something is really wrong.");
throw ex;

try {
  y = 0;
  x = 10 / y;
} catch (Exception ex) {
  System.out.println(ex.getMessage());
} finally {
  // Code that always gets executed
}
```

```
Exception up = new Exception("Something is really wrong.");
throw up;  // ha ha

try {
  y = 0;
  x = 10 / y;
} catch (Exception ex) {      // Variable "ex" is optional
  Console.WriteLine(ex.Message);
} finally {
  // Code that always gets executed
}
```

Java	Namespaces	C#

```
package harding.compsci.graphics;
```

```
namespace Harding.Compsci.Graphics {
  ...
}

or

namespace Harding {
  namespace Compsci {
    namespace Graphics {
```

```
    ...
  }
 }
}
```

```
// Import single class
using Rectangle = Harding.CompSci.Graphics.Rectangle;

// Import all class
using Harding.Compsci.Graphics;
```

```
// Import single class
import harding.compsci.graphics.Rectangle;

// Import all classes
import harding.compsci.graphics.*;
```

Java Classes / Interfaces C#

```
// Accessibility keywords
public
private
protected
static
```

```
// Accessibility keywords
public
private
internal
protected
protected internal
static
```

```
// Inheritance
class FootballGame extends Competition {
  ...
}

// Interface definition
interface IAlarmClock {
  ...
}

// Extending an interface
interface IAlarmClock extends IClock {
```

```
// Inheritance
class FootballGame : Competition {
  ...
}

// Interface definition
interface IAlarmClock {
  ...
}

// Extending an interface
interface IAlarmClock : IClock {
```

```
  ...
}

// Interface implementation
class WristWatch implements IAlarmClock, ITimer {
  ...
}
```

```
  ...
}

// Interface implementation
class WristWatch : IAlarmClock, ITimer {
  ...
}
```

Java	Constructors / Destructors	C#

```
class SuperHero {
  private int mPowerLevel;

  public SuperHero() {
    mPowerLevel = 0;
  }

  public SuperHero(int powerLevel) {
    this.mPowerLevel= powerLevel;
  }

  // No destructors, just override the finalize method
  protected void finalize() throws Throwable {
    super.finalize();   // Always call parent's finalizer
  }
}
```

```
class SuperHero {
  private int mPowerLevel;

  public SuperHero() {
    mPowerLevel = 0;
  }

  public SuperHero(int powerLevel) {
    this.mPowerLevel= powerLevel;
  }

  ~SuperHero() {
    // Destructor code to free unmanaged resources.
    // Implicitly creates a Finalize method.
  }
}
```

Java	Objects	C#

```java
SuperHero hero = new SuperHero();

hero.setName("SpamMan");
hero.setPowerLevel(3);

hero.Defend("Laura Jones");
SuperHero.Rest();  // Calling static method

SuperHero hero2 = hero;   // Both refer to same object
hero2.setName("WormWoman");
System.out.println(hero.getName());  // Prints WormWoman

hero = null;   // Free the object

if (hero == null)
  hero = new SuperHero();

Object obj = new SuperHero();
System.out.println("object's type: " + obj.getClass().toString());
if (obj instanceof SuperHero)
  System.out.println("Is a SuperHero object.");
```

```c#
SuperHero hero = new SuperHero();

hero.Name = "SpamMan";
hero.PowerLevel = 3;

hero.Defend("Laura Jones");
SuperHero.Rest();   // Calling static method

SuperHero hero2 = hero;   // Both refer to same object
hero2.Name = "WormWoman";
Console.WriteLine(hero.Name);   // Prints WormWoman

hero = null ;   // Free the object

if (hero == null)
  hero = new SuperHero();

Object obj = new SuperHero();
Console.WriteLine("object's type: " + obj.GetType().ToString());
if (obj is SuperHero)
  Console.WriteLine("Is a SuperHero object.");
```

Java	Properties	C#

```java
private int mSize;

public int getSize() { return mSize; }
public void setSize(int value) {
  if (value < 0)
    mSize = 0;
  else
    mSize = value;
}
```

```c#
private int mSize;

public int Size {
  get { return mSize; }
  set {
    if (value < 0)
      mSize = 0;
    else
      mSize = value;
```

```
int s = shoe.getSize();
shoe.setSize(s+1);
```

```
  }
}

shoe.Size++;
```

Java	Structs	C#

No structs in Java.

```
struct StudentRecord {
  public string name;
  public float gpa;

  public StudentRecord(string name, float gpa) {
    this.name = name;
    this.gpa = gpa;
  }
}

StudentRecord stu = new StudentRecord("Bob", 3.5f);
StudentRecord stu2 = stu;

stu2.name = "Sue";
Console.WriteLine(stu.name);    // Prints "Bob"
Console.WriteLine(stu2.name);   // Prints "Sue"
```

Java	Console I/O	C#

```
java.io.DataInput in = new java.io.DataInputStream(System.in);
System.out.print("What is your name? ");
String name = in.readLine();
System.out.print("How old are you? ");
int age = Integer.parseInt(in.readLine());
System.out.println(name + " is " + age + " years old.");
```

```
Console.Write("What's your name? ");
string name = Console.ReadLine();
Console.Write("How old are you? ");
int age = Convert.ToInt32(Console.ReadLine());
Console.WriteLine("{0} is {1} years old.", name, age);
// or
Console.WriteLine(name + " is " + age + " years old.");
```

```
int c = System.in.read();   // Read single char
System.out.println(c);      // Prints 65 if user enters "A"
```

```
int c = Console.Read();  // Read single char
Console.WriteLine(c);    // Prints 65 if user enters "A"
```

```
// The studio costs $499.00 for 3 months.
System.out.printf("The %s costs $%.2f for %d months.%n",
"studio", 499.0, 3);
```

```
// The studio costs $499.00 for 3 months.
Console.WriteLine("The {0} costs {1:C} for {2} months.\n", "studio",
499.0, 3);
```

```
// Today is 06/25/04
System.out.printf("Today is %tD\n", new java.util.Date());
```

```
// Today is 06/25/2004
Console.WriteLine("Today is " +
DateTime.Now.ToShortDateString());
```

Java	File I/O	C#

```
import java.io.*;

// Character stream writing
FileWriter writer = new FileWriter("c:\\myfile.txt");
writer.write("Out to file.\n");
writer.close();

// Character stream reading
FileReader reader = new FileReader("c:\\myfile.txt");
BufferedReader br = new BufferedReader(reader);
String line = br.readLine();
while (line != null) {
  System.out.println(line);
  line = br.readLine();
```

```
using System.IO;

// Character stream writing
StreamWriter writer = File.CreateText("c:\\myfile.txt");
writer.WriteLine("Out to file.");
writer.Close();

// Character stream reading
StreamReader reader = File.OpenText("c:\\myfile.txt");
string line = reader.ReadLine();
while (line != null) {
  Console.WriteLine(line);
  line = reader.ReadLine();
```

```
}
reader.close();

// Binary stream writing
FileOutputStream out = new FileOutputStream("c:\\myfile.dat");
out.write("Text data".getBytes());
out.write(123);
out.close();

// Binary stream reading
FileInputStream in = new FileInputStream("c:\\myfile.dat");
byte buff[] = new byte[9];
in.read(buff, 0, 9);   // Read first 9 bytes into buff
String s = new String(buff);
int num = in.read();   // Next is 123
in.close();
```

```
}
reader.Close();

// Binary stream writing
BinaryWriter out = new
BinaryWriter(File.OpenWrite("c:\\myfile.dat"));
out.Write("Text data");
out.Write(123);
out.Close();

// Binary stream reading
BinaryReader in = new
BinaryReader(File.OpenRead("c:\\myfile.dat"));
string s = in.ReadString();
int num = in.ReadInt32();
in.Close();
```