
SWINBURNE UNIVERSITY OF TECHNOLOGY

# COS20007

## 10.2D: An In-Depth comparison of Java and C Sharp

(102573957) Matthew Coulter

# Java and Object Orientated Programming

Like C Sharp, Java is an objected orientated language. This means that all of its core features (polymorphism, encapsulation, abstraction and inheritance) can be implemented in one way or another. For each of these concepts, screenshots highlight the key differences and/or similarities for the implementation of both C Sharp and Java. A conclusion will thoroughly explain these details, and revealing the opportunities or limitations of Java.

## Polymorphism

To demonstrate a comparison of polymorphism in both Java and C Sharp, I will be referencing our shape drawing program. Simply, Polymorphism is about being able to treat a child object as its parents object. In this demonstration, this is evident as a rectangle and a circle will be added to an array of type shape. Then, in a foreach loop, this shape.Draw() method is called.

### C Sharp

```
using System;
using System.Collections.Generic;

namespace ShapeCSharp
{
    0 references
    public class Program
    {
        0 references
        static void Main()
        {
            List<Shape> shapes = new List<Shape>();

            shapes.Add(new Rectangle(0, 0, 50, 100));
            shapes.Add(new Circle(100, 100, 20));

            foreach (Shape shape in shapes)
            {
                shape.Draw(shape.X, shape.Y);
            }
        }
    }
}
```

### Java

```
import java.util.ArrayList;
import java.util.List;

public class Main {
    public static void main() {
        List<Shape> shapes = new ArrayList<>();

        shapes.add(new Rectangle(0,0,50,100));
        shapes.add(new Circle(100,100,20));

        for (Shape shape : shapes)
        {
            shape.Draw(shape.get_x(),shape.get_y());
        }

    }
}
```

## Conclusion

Both languages implement this abstract code in the same way. And fundamentally, they should because they are both object orientated languages. It should be noted that in Java, the List is declared as an ArrayList<>(), there is no capitalisation on '.add' and the foreach loop is structured as 'for (type var : array) { }'. These are the minor syntax differences.

# Abstraction

Abstraction is a process apart of achieving polymorphism. This requires setting an abstract class, declaring abstract methods and overriding them in corresponding child classes. In this section we'll will explore how Java implements this compared to C Sharp.

## C Sharp

Declaring Abstract Class Shape.cs

```
public abstract class Shape
{
```

Declaring Abstract method in Shape.cs

```
public abstract void Draw(float x, float y);
```

Overriding Abstract Method in Rectangle.cs

```
public override void Draw(float x, float y)
{
    //Draw Method Here...
}
```

## Java

Declaring Abstract Class in Shape.java

```
public abstract class Shape {
```

Declaring abstract method in Shape.java

```
public abstract void Draw(float x, float y);
```

Overriding abstract method in Rectangle.java

```
@Override
public void Draw(float x, float y)
{
    //draw function here
}
```

## Conclusion

The implementation of abstraction is very similar in both C Sharp and Java. The syntax is identical for the declaration of an abstract class and abstract method. The only difference is the syntax when overriding. Both languages require all of the abstract methods to be overwritten.

## Inheritance

Inheritance is the other part in order to achieve polymorphism. This section demonstrates in both languages how a child class can inherit a parent class and initialise both child and parent class variables.

### C Sharp

Inheriting a parent class

```
public class Rectangle : Shape
{
```

Passing initialisation variables into the parent class

```
public Rectangle(float x, float y, float width, float height) : base(x,y)
{
    _width = width;
    _height = height;
}
```

### Java

Inheriting a parent class

```
public class Rectangle extends Shape {
```

Passing initialisation variables into the parent class

```
public Rectangle(float x, float y, float width, float height)
{
    super(x,y);
    _width = width;
    _height = height;
}
```

### Conclusion

Both C Sharp and Java allow for the same possibilities when inheriting, the only difference is the syntax. C Sharp utilises the ':' characters to extends the child class to the parents class whereas Java writes the word extends. When passing variables through the child class into the bass class, C Sharp uses ': base(variables)' whereas Java writes 'super(variables)' in the body of the constructor.

## Encapsulation

Encapsulation is used in order to make local variables accessible from child classes and through object references. This section is dedicated to comparing how both languages implement properties to utilise the concept of encapsulation.

### C Sharp

Property declaration

```
private int _count = 0;
0 references
public int Count
{
    get { return _count; }
    set { _count = value; }
}
```

**OR**

```
public int Count { get => _count; set => _count = value; }
```

### Java

Property declaration

```
private int _count = 0;

public void setCount(int value)
{
    _count = value;
}

public int getCount()
{
    return _count;
}
```

### Conclusion

When we initialise variables, both C Sharp and Java have the accessibility modifier prefaced. However, the Java method for implementing properties is more manual than that of C Sharp. In C Sharp, the getting and setting is automatic because you don't need to establish the set type as void and arguments. In can both be done within the same body.

# Syntax

Whilst there are many similarities and differences and syntax, I don't believe this to be one of the key considerations of a language. It is much more about what can be achieved in one language that can't be achieved in others, or "how easy is it to achieve". Hence, this list will not be so extensive and only cover some basics.

## Logical Operators

All of the logic operators are the same between both languages. Here are a few examples.

| Operator | C Sharp | Java |
|---|---|---|
| **And** | && | && |
| **Or** | \|\| | \|\| |
| **Not** | ! | ! |
| **Equal to** | == | == |
| **Not Equal to** | != | != |
| **Greater Than** | > | > |
| **Less Than** | < | < |
| **Greater than or equal to** | >= | >= |

# Conclusion

Now after looking through all of the above, besides expected minor syntax differences, you are now probably still wondering, what is the big functional difference between Java and C Sharp? The answer is simply not much. They both can achieve the same thing, and as they are both object-orientated languages, the process is very similar for both. Yes, you may come across a few hiccups if you are switching to java from C Sharp or C Sharp to java does not support the 'goto' statement in a switch case, noticing that there are no implicit property declarations in java, C Sharp only supports checked exceptions and java. However, the bigger picture lies barely within the code, but instead externally. This comes down to IDEs, frameworks, community support, purpose and standards.

Interestingly enough, as these languages are currently the 2 most popular languages, both currently have lots of support for implementing third-party libraries. This also means there is a sufficient amount of IDEs for all programmers. Since Java operates of the Java SE Development Kit (JDK), an appropriate version needs to be installed, and likewise for C Sharps .net framework. If we analyse which language has been used for popular applications nowadays, Java has been used for more applications in the data field such as Facebook, Amazon, eBay And LinkedIn whereas C Sharp has been predominantly used for user applications and games through its collaboration with Unity. These implementations have created the standards for what each should be used for, but in no means prevents functionality of either languages.