C# / Java Language Comparison

Robert Bachmann

April 4, 2011

Outline

- My favorite pitfalls
- Common conventions
- Type system
- Generics
- Keywords
- Exceptions
- Specific features of Java
- Specific features of C#

My favorite pitfalls

- `str == "hello"`
  Checks equality in C#. Checks identity in Java.
- Virtual methods
  Opt-out in Java (`final` keyword) vs. opt-in in C# (`virtual` keyword)
- Dates
  `new GregorianCalendar(2011, 3, 4)`[1] vs.
  `new DateTime(2011,4,4)`

[1] Hint: use YodaTime, Date4J, …

Common conventions

- Package names / Namespaces: `org.acme.foo` vs. `Acme.Foo`
- Interfaces: `Supplier` vs. `ISupplier`
- Methods: `doSomething` vs. `DoSomething`
- Opening braces: Same line in Java vs. next line in C#

Type system

Basic types

Java	C#	CLR Type
String	string	String
Object	object	Object
boolean	bool	Boolean
char	char	Char
short	short	Int16
int	int	Int32
long	long	Int64
double	double	Double
float	float	Single
byte	sbyte	SByte
—	byte	Byte
—	ushort	UInt16
—	uint	UInt32
—	ulong	UInt64

Type system

- Java has reference and primitive types.
- C# has reference and value types.
  Uses an unified type system, for example you can do `42.ToString()`
  It's possible to create custom value types:

```
public struct Point
{
  public int X { get; set; }
  public int Y { get; set; }

  public override string ToString()
  {
    return String.Format("({0},{1})", X, Y);
  }
}
```

Generics

- Java uses *type erasure*. Generics are (mostly) a compiler feature.

  ```
  new ArrayList<Integer>().getClass();
  // -> ArrayList
  new ArrayList<String>().getClass();
  // -> ArrayList
  ```

- C# uses *reified generics*. Generics are a runtime feature.

  ```
  new List<int>().GetType();
  // -> List`1[Int32]
  new List<string>().GetType();
  // -> List`1[String]
  ```

Generics in Java

Code:

```java
ArrayList<String> list1 = new ArrayList<String>();
list1.add("hello");
String s = list1.get(0);

ArrayList<Integer> list2 = new ArrayList<Integer>();
list2.add(42);
int i = list2.get(0);
```

Decompiled:

```java
ArrayList list1 = new ArrayList();
list1.add("hello");
String s = (String)list1.get(0);

ArrayList list2 = new ArrayList();
list2.add(Integer.valueOf(42));
int i = ((Integer)list2.get(0)).intValue();
```

Generics in C#

Code:

```
List<string> list1 = new List<string>();
list1.Add("hello");
string s = list1[0];

List<int> list2 = new List<int>();
list2.Add(42);
int i = list2[0];
```

Decompiled:

```
List<string> list = new List<string>();
list.Add("hello");
string text = list[0];

List<int> list2 = new List<int>();
list2.Add(42);
int num = list2[0];
```

Generics

List factory method example

```
public class ListFactory {
  public static <T> List<T> newList() {
              //^^^
    return new ArrayList<T>();
  }
}

// use with
List<String> list = ListFactory.newList();
```

Generics

List factory method example

```
public class ListFactory
{
  public static IList<T> newList<T>()
  {                             //^^^
    return new List<T>();
  }
}

// use with
IList<int> list = ListFactory.newList<int>();

// Does not work:
IList<int> list = ListFactory.newList();
// The type arguments for method
// ListFactory.newList<T>()' cannot be inferred
// from the usage.
// Try specifying the type arguments explicitly.
```

Generics

Constraints

```java
<T extends Comparable<T>> T max(T a, T b) {
    if (a.compareTo(b) > 0)
        return a;
    else
        return b;
}
```

```csharp
T max<T>(T a, T b)
  where T : IComparable<T>
{
  if (a.CompareTo(b) > 0)
    return a;
  else
    return b;
}
```

Generics

Constraints

- `where T : ClassOrInterface`,
  same as `T extends ClassOrInterface`
- `where T : new()`, T has a parameterless constructor.
- `where T : class`, T is a reference type.
- `where T : struct`, T is a value type.

Keywords

- Visibility

Common:	`public`, `protected`, and `private`
Java:	Package visibility.
C#:	`internal` and `protected internal`

- `final` vs. `sealed` (for methods and classes)
- `final` vs. `readonly` (for members)
- `for (T item : items)` vs. `foreach (T item in items)`
- `instanceof` vs. `is`
- `Foo.class` vs. `typeof(Foo)`

Exceptions

- Java
  - Must throw a `Throwable`
  - Has checked and unchecked exceptions
  - Re-throw with `throw e;`
- C#
  - Must throw an `Exception`
  - Has no checked exceptions
  - Re-throw with `throw;`
    `throw e;` has a different semantic

Specific features of Java

- Anonymous inner classes
- static import

Specific features of C#

- First class properties
- `using` blocks
- Preprocessor
- Delegates and lambda expressions
- LINQ

Specific features of C#

using blocks

```
// using (IDisposable)
static String ReadFirstLineFromFile(String path) {
  using (StreamReader reader = File.OpenText(path)) {
    return reader.ReadLine();
  }
}
```

```
// http://download.java.net/jdk7/docs/technotes/
//  guides/language/try-with-resources.html
// try (AutoClosable)
static String readFirstLineFromFile(String path)
  throws IOException {
  try (BufferedReader br =
    new BufferedReader(new FileReader(path)) {
    return br.readLine();
  }
}
```

Specific features of C#

Preprocessor

```
        public static void Main(string[] args)
        {
#if DEBUG
            Console.WriteLine("Debug␣...");
#endif
            Console.WriteLine("Hello");
        }
```

Specific features of C#

Delegates and lambda expressions

```
public delegate bool Predicate<T>(T obj);

public static void Main(string[] args)
{
  List<string> list = new List<string>{
    "Hello", "World", "what's", "up"
  };
  Predicate<String> shortWord;
  shortWord = delegate(string s) {return s.Length < 4;};
  shortWord = s => s.Length < 4;
  list.Find(shortWord);
  // or just
  list.Find(s => s.Length < 4);
}
```

Specific features of C#

LINQ

```
public static void Main(string[] args)
{
  var list = new List<string> {
    "Hello", "World", "what's", "up"
  };

  var result = from item in list select item.Length;

  foreach (int i in result) {
    Console.WriteLine(i);
    // -> 5, 5, 6, 2
  }
}
```

Specific features of C#

More features

- Operator overloading
- `dynamic` pseudo-type
- `yield` keyword, similar to Python
- Extension methods
- Expression trees
- Explicit interfaces
- Partial classes
- …
- See http://en.wikipedia.org/wiki/Comparison_of_C_Sharp_and_Java