Java and C# in Depth

Carlo A. Furia, Marco Piccioni, Bertrand Meyer

Exercise Session – Week 2
Organizational Part

• Send your assignments to alexey.kolesnichenko@inf.ethz.ch

• Start teaming up! Teams are of 3 people

• Enroll to piazza, if you haven’t: piazza.com/ethz.ch/spring2014/252028400l/home
Agenda

- Quizzes
- More quizzes
- And even more quizzes ...
Quiz 1. What will be printed?

Integers

Java and C# in Depth -- Exercise

- (Java) Integer and int are different types
  - Autoboxing and unboxing
    - JVM creates a new object, sometime reuses an old one.
- (C#) no caching of boxed objects
Quiz 2. What will be printed?

Floating point numbers

```java
public class Program{
    public static void main(String[] args) {
        double d1=4.0;
        double d2=3.1;
        System.out.println(d1 - d2);
    }
}
```

```
public class Program {
    static void Main(string[] args) {
        double d1 = 4.0;
        double d2 = 3.1;
        Console.WriteLine(d1 - d2);
    }
}
```

- 0.8999999999999999
- 0.9

- Why the imprecision?
  - Nature of floating point numbers (IEEE 754)

- Why the precision?
  - Don’t get fooled! C# rounds float/double values to certain significant digits for base-10 string representation (what you see is not what it is)
  - Using `BigDecimal` in Java and `decimal` in C# solves these problems. Both classes work on base-10 representations, and are therefore slow.
Standard Numeric Format Strings (C#)

```csharp
float sum = 0;
for (int i = 1; i < 11; i++){
    sum += 0.1f;
    Console.Write(String.Format("{0,-10}", sum.ToString()));
    Console.Write(",");
    Console.WriteLine(sum.ToString("g9"));
}
```

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.100000001</td>
</tr>
<tr>
<td>0.2</td>
<td>0.200000003</td>
</tr>
<tr>
<td>0.3</td>
<td>0.300000012</td>
</tr>
<tr>
<td>0.4</td>
<td>0.400000006</td>
</tr>
<tr>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>0.6</td>
<td>0.600000024</td>
</tr>
<tr>
<td>0.7</td>
<td>0.700000048</td>
</tr>
<tr>
<td>0.8000001</td>
<td>0.800000072</td>
</tr>
<tr>
<td>0.9000001</td>
<td>0.900000095</td>
</tr>
<tr>
<td>1</td>
<td>1.00000012</td>
</tr>
</tbody>
</table>
Quiz 3: What does it do?

✓ Does the following Java code test oddity correctly?

```java
public static boolean isOdd(int i) {
    return i % 2 == 1;
}
```

No

✓ What do you get from the following C# code?

```csharp
public int Divide(int operand1, int operand2){
    return operand1 / operand2;
}
public int Test(){
    return Divide(4, 0);
}
```

DivideByZeroException

✓ What if we change int’s to double’s?

Compilation error (Java)
Double.PositiveInfinity or Double.NegativeInfinity (C#)
Quiz 4. What will be printed?

Strings

- Strings are immutable objects in both languages
- Operator overloading in C#
  - Reference equality

```java
public class ImmutableStrings{
    public static void main(String[] args){
        String a = "string";
        String b = "string";
        String c = "str";
        c += "ing";
        System.out.println(a.equals(b));
        System.out.println(a == b);
        System.out.println(a == c);
    }
}
true
true
false
```

```csharp
class ImmutableStrings {
    static void Main(string[] args) {
        string a = "string";
        string b = "string";
        string c = "str";
        c += "ing";
        Console.WriteLine(a.Equals(b));
        Console.WriteLine(a == b);
        Console.WriteLine(a == c);
    }
}
true
true
true
```
<table>
<thead>
<tr>
<th># of appends</th>
<th>+</th>
<th>StringBuilder</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.000</td>
<td>0.00</td>
</tr>
<tr>
<td>100</td>
<td>0.000</td>
<td>0.00</td>
</tr>
<tr>
<td>1,000</td>
<td>0.000</td>
<td>0.00</td>
</tr>
<tr>
<td>2,500</td>
<td>0.000</td>
<td>0.00</td>
</tr>
<tr>
<td>5,000</td>
<td>0.020</td>
<td>0.00</td>
</tr>
<tr>
<td>7,500</td>
<td>0.050</td>
<td>0.00</td>
</tr>
<tr>
<td>10,000</td>
<td>0.090</td>
<td>0.00</td>
</tr>
<tr>
<td>15,000</td>
<td>0.250</td>
<td>0.00</td>
</tr>
<tr>
<td>25,000</td>
<td>1.052</td>
<td>0.00</td>
</tr>
<tr>
<td>35,000</td>
<td>2.373</td>
<td>0.00</td>
</tr>
<tr>
<td>50,000</td>
<td>5.699</td>
<td>0.00</td>
</tr>
<tr>
<td>65,000</td>
<td>10.625</td>
<td>0.00</td>
</tr>
<tr>
<td>75,000</td>
<td>14.831</td>
<td>0.01</td>
</tr>
<tr>
<td>85,000</td>
<td>19.418</td>
<td>0.01</td>
</tr>
<tr>
<td>100,000</td>
<td>27.159</td>
<td>0.01</td>
</tr>
<tr>
<td>150,000</td>
<td>65.374</td>
<td>0.01</td>
</tr>
<tr>
<td>250,000</td>
<td>209.221</td>
<td>0.02</td>
</tr>
<tr>
<td>350,000</td>
<td>441.615</td>
<td>0.02</td>
</tr>
<tr>
<td>500,000</td>
<td>910.129</td>
<td>0.04</td>
</tr>
<tr>
<td>650,000</td>
<td>1521.708</td>
<td>0.06</td>
</tr>
<tr>
<td>750,000</td>
<td>1999.305</td>
<td>0.06</td>
</tr>
<tr>
<td>850,000</td>
<td>2576.575</td>
<td>0.06</td>
</tr>
<tr>
<td>1,000,000</td>
<td>3562.933</td>
<td>0.07</td>
</tr>
</tbody>
</table>
Quiz 5. Can we do this?

Switch

```java
int i1;
i1 = 2;
// can we do this?
switch (i1) {
    case 1: print(1);
    case 2: print(2);
    case 3: print(3);
}
```

```java
String s1;
...
// or this?
switch (s1) {
    case "1": ...
    case "2": ...
    case "3": ...
}
```

- **Rules about “fall-through”**
- **Data types**
  - (Java) **char**, **byte**, **short**, **int**, their corresponding reference types, **String**, or **enum**
  - (C#) Numeric, **String**, or enumeration data types
    - Refer to the language specification for more information
Quiz 6. What do these programs do? (Java)

➢ Type

public class Type1 {
    public static void main(String[] args) {
        String s = null;
        System.out.println(s instanceof String);
    }
}

public class Type2 {
    public static void main(String[] args) {
        System.out.println(new Type2() instanceof String);
    }
}

public class Type3 {
    public static void main(String args[]) {
        Type3 t3 = (Type3) new Object();
    }
}
Quiz 7. What will be printed? (Java)

```java
public class Test {
    int x = 5;

    public static void main(String[] args) {
        final Test f1 = new Test();
        Test f3 = testSwitch (f1);
        System.out.println((f1 == f3) + " " + (f1.x == f3.x) + " " + f1.x + " " + f3.x);
    }

    static Test testSwitch(final Test x) {
        final Test z = x;
        z.x = 6;
        return z;
    }
}
```

References f1, z, and f3 refer to the same instance.
- final assures that a reference variable cannot be referred to a different object.
- But final doesn’t keep the object’s state from changing.
Quiz 8. Type var and dynamic in C#

Example

```csharp
var myData = 365;
```

True or False?

- The `var` keyword can be used to define local variables in a method or property scope, return values, parameters, or field data of a custom type;  
  - True (False marked)

- Implicitly typed variables could be declared and initialized at different locations;
  - True (False marked)

- Given the above declaration, assignment `myData = false;` will change the type of `myData` from `int` to `bool.`
  - True (False marked)

What if we change `var` to `dynamic`?

When to use?

- `var`: Mostly, for results from LINQ queries
- `dynamic`: scripting, interfacing with other languages, etc.
Quiz 9. Nullable data types in C#

- A nullable type can represent all the values of its underlying type, plus the value null

```csharp
int? nullableInt = 10;
nullableInt = null;
```

- True or False?
  - Nullable types could be based on reference types too, e.g.
    ```csharp
    String? nullableStr = “A”;
    ```
    - False
  - The ? suffix notation is a shorthand for creating an instance of the generic System.Nullable<T> structure type.
    ```csharp
    if (nullableInt.HasValue)
        Console.WriteLine("Value of 'nullableInt' is: {0}",
                        nullableInt.Value);
    aInt = nullableInt.HasValue ? nullableInt.Value : 100;
    // Could be written as:
    aInt = nullableInt ?? 100;
    ```
    - True
enum

```
using System;
class Program{
    private static int[] resource = new int[] {0, 1, 2};
    public enum Size{Small, Medium, Large}

    public static void Method1(Size theSize){
        Console.WriteLine(theSize);
        Console.WriteLine("Resource: {0}", resource[(int)theSize]);
    }

    static void Main(string[] args){
        Method1(Size.Small);
        Method1(Size.Large);
        Method1((Size) 1);
        Method1((Size) 3);
    }
}
```

- Lacks type-safety
- Problem with serialization
Questions?