Java and C# in Depth
Carlo A. Furia, Marco Piccioni, Bertrand Meyer

Exercise Session – Week 4
Don’t forget to form project groups by tomorrow (March 11th, 2014)
Please do not submit assignments that do not compile
Agenda

- Quizzes

- More quizzes

- And even more quizzes ...
Quiz 1. Differences between Struct and Class (C#)

- Structs define value types, while classes define reference types.
- A struct cannot inherit from another struct or from classes.
- A struct can only be used as the base for a struct, but not for a class.
  - A struct cannot be used as the base.
  - A struct can implement interfaces.
- A default constructor will be provided for a struct, only if it does not have any user-defined constructors.
  - A struct always has a default constructor, which clears the memory to zeroes.
  - Thus, although a struct may declare constructors, those constructors *must* take at least one argument.
- The struct members cannot have initializers.
Quiz 2. Abstract Classes (Java Vs. C#)

- Can an abstract class have no abstract methods?
  - (Java) Yes.
  - (C#) Yes.

- Can an abstract class have more than one superclass?
  - (Java) No, single inheritance only.
  - (C#) No, single inheritance only.

- Can an abstract class be a subclass of a concrete class?
  - (Java) Yes, e.g. class Object.
  - (C#) Yes, e.g. class Object.
Quiz 3. Code Organization (Java Vs. C#)

- How many package or namespace declarations may be contained in one source file?
  - (Java) One at most.
  - (C#) No restriction.

- How is a package/namespace name related with the physical storage structure of code?
  - (Java) Package names correspond to the directory names.
  - (C#) No relation.

- How many classes can be contained in one source file?
  - (Java) At most one public class, but no restrictions otherwise.
  - (C#) No restriction.
Quiz 4. What does the program do?

Static method

```java
public class Null {
    public static void greet() {
        System.out.println("Hello world!");
    }
    public static void main(String[] args) {
        ((Null) null).greet();
    }
}
```

A qualifying expression for a static method invocation is evaluated, but its value is ignored.

```java
class Null{
    static void greet() {
        Console.WriteLine("Hello world!");
    }
    static void Main(string[] args){
        ((Null) null).greet();
    }
}
```

Compilation error!
Member 'Null.greet()' cannot be accessed with an instance reference; qualify it with a type name instead.
Quiz 5: Overloading

- Is it ok to have the following method declarations in a class A? Why?

```java
void print(int i){...}  // 1
void print(float f){...} // 2
int print(float f){...}  // 3
```

1 and 2: Fine.
1 and 3: Fine.
2 and 3: Error.

- If class A has the following two declarations,

```java
void print(int i){...}  // 1
void print(float f){...} // 2
```

and in class B, a subclass of A, we define two methods as follows, will it be ok? Why?

```java
void print(long i){...}   // 3
void print(int f){...}    // 4
```

1, 2, and 3: Fine (overloading)
1, 2, and 4: Java: also fine (overriding); C#: warning (declare as new)
Quiz 6. What does the program do?

Method overloading

```java
public class Base{
    public virtual void M1(double val){
        Console.WriteLine("Base.M1(double)">Your answer goes here.
    }
}

public class Derived : Base{
    public virtual void M1(int val){
        Console.WriteLine("Derived.M1(int)">Your answer goes here.
    }
}

class Test{
    static void Main(string[] args){
        Derived d = new Derived();
        Base b = d;
        b.M1(3);
        d.M1(3);
    }
}
```

```
Base.M1(double)
Derived.M1(int)
```
Quiz 7. What does the program do?

Method overriding (Java)

```java
import java.util.*;
public class Name {
    private final String first, last;
    public Name(String first, String last) {
        this.first = first;
        this.last = last;
    }
    public boolean equals(Object o) {
        if (!(o instanceof Name))
            return false;
        Name n = (Name) o;
        return n.first.equals(first) && n.last.equals(last);
    }
    public static void main(String[] args) {
        Set<Name> s = new HashSet<Name>();
        s.add(new Name("Mickey", "Mouse"));
        System.out.println(
            s.contains(new Name("Mickey", "Mouse")));
```
Anonymous function expressions: C# (1)

Anonymous method expressions

delegate void Printer(string s);

class TestClass{
    static void DoWork(string k){
        System.Console.WriteLine(k);
    }
    static void Main(){
        Printer p = TestClass.DoWork;
        // p = new Printer(TestClass.DoWork);
        p("Delegate with named method.");

        p = delegate (string j){
            System.Console.WriteLine(j);
        };
        p("Delegate with anonymous method.");
    }
}

Anonymous function expressions: C# (2)

- Lambda expressions
  - Statement lambda
    
    Arguments => {Statements}
    
    (int i) => {
        bool isEven = (i%2 == 0);
        return isEven;
    }

  - Expression lambda
    
    Arguments => Expression
    
    (int i) => (i % 2) == 0
    
  - Could also be used to construct expression tree objects

    Expression<Func<int, int>> exp = (n) => (n * 2 + 1) * 4;

- Arguments could be implicitly typed

- Parentheses are optional for single argument but not in the case of no argument

    Func<int, int> Double = (n) => n*2;

    i => (i % 2) == 0

    () => {Console.Write ("...");}
Variables in anonymous functions: C#

- An anonymous function can access the local variables and (some of) the parameters of the enclosing method (called outer variables)
  - Value parameters, and parameter array
    - In an instance function member of a class, the `this` value is considered a value parameter
  - Not `ref` or `out` parameters of the enclosing method

- Defining local variables
  - **Can** declare local variables with the same name as outer class member variables.
  - **Cannot** have a local variable with the same name as a local variable in the enclosing method.
**Quiz 8. What will be printed?**

- **Anonymous method expressions**

```csharp
delegate void D();

static D[] F() {
    D[] result = new D[3];
    int i;
    for (i = 0; i < 3; i++) {
        result[i] = () => {
            Console.WriteLine(i);
        };
    }
    return result;
}

static void Main() {
    foreach (D d in F()) d();
}
```


- **Anonymous method expressions**

```csharp
delegate void D();

static D[] F() {
    D[] result = new D[3];
    int i;
    for (i = 0; i < 3; i++) {
        int j = i;
        result[i] = () => {
            Console.WriteLine(j);
        };
    }
    return result;
}

static void Main() {
    foreach (D d in F()) d();
}
```

| 0 | 1 | 2 |
Questions?