Konzepte objektorientierter Programmierung

Prof. Dr. Peter Müller

Werner Dietl

Software Component Technology

Exercises 3: Some More OO Languages

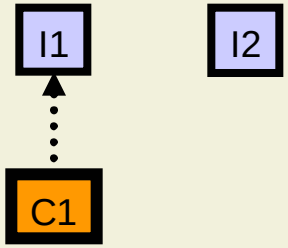
Wintersemester 06/07

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Agenda for Today

- Homework
- Finish Java Overview
- Quick look at C++ and C#

Homework – Exercise 1

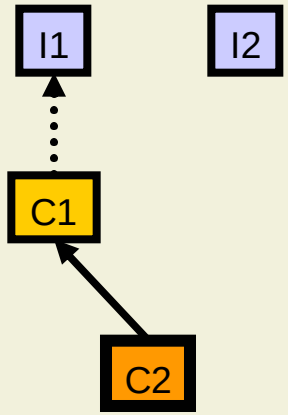



```
interface I1 {}
interface I2 {}
final class C1 implements I1 {}

public class Ex1 {
    public static void main( String[] args ) {
        C1 c1 = new C1();
        I2 i2 = (I2) c1;
    }
}
```

- Compilation Error: inconvertible types

Homework – Exercise 2




```
interface I1 {}
interface I2 {}
class C1 implements I1 {}
final class C2 extends C1 {}

public class Ex2 {
  public static void main( String[] args ) {
     C1 c1 = new C2();
     I2 i2 = (I2) c1;
  }
}
```

- Runtime Error: java.lang.ClassCastException

Homework – Exercise 3

```
class C1 {
   int age = 99;
   int getAge() {
     return age; }
}

class C2 extends C1 {
   int age = 22;
   int getAge() {
     return age; }
}
```

- Output:

```
c1.age:          99
c1.getAge():     99
c2.age:          22
c2.getAge():     22
c1=c2.age:       99
c1=c2.getAge(): 22
```

CAT calls in Eiffel

- Changed Availability or Type
- Removing a feature from the interface
- Using a subtype for a parameter in an overriding feature
- Behavior not specified
- Depends on the Eiffel compiler used

```
class SUPER

feature
  f is
  do
     print("Hello World!%N")
  end

  g(p: SUPER) is
  do
     print("g in SUPER%N")
  end
end -- class SUPER
```

```
class SUB

inherit SUPER undefine f redefine g end
feature {NONE}
  f is do
     print("My private message%N")
  end

feature
  g( p: SUB ) is do
     print("g in SUB%N")
     p.subm
  end
  subm is do
     print("New feature of SUB called!%N")
  end
```

Using the classes

```
local
        sup:  SUPER
        sup2: SUPER
        sub:  SUB
do
        create sub
        sup := sub
        sup.f
        sup.g( sup )
        create sup2
        sup.g( sup2 )
end
```

Overriding Example – Setup

```
class Upper {}
class Middle extends Upper {}
class Lower extends Middle {}

class Super {
  void foo( Middle a1 ) {
    System.out.println(
      "Super.foo("+a1+")");
  }
}
class Sub extends Super {
  void foo( Upper a1 ) {
    System.out.println(
      "Sub.foo("+a1+")");
  }
}
```

```
Super super1;
Sub   sub1;
Lower lower1 =
    new Lower();
Upper upper1 =
    new Upper();
```

Overriding Example – Main

```
Super: foo(Middle a1)
Sub:   foo(Upper a1)
```

```
System.out.println("Calls on Super object:");
      super1 = new Super();
  1   super1.foo( lower1 );
  2   super1.foo( upper1 );

System.out.println("\nCalls on Sub object in
  Super reference:");
      super1 = new Sub();
  3   super1.foo( lower1 );
  4   super1.foo( upper1 );

System.out.println("\nCalls on Sub object in
  Sub reference:");
      sub1 = new Sub();
  5   sub1.foo( lower1 );
  6   sub1.foo( upper1 );
```

Compilation Error!

Compilation Error!

Overriding Example - Output

```
Super: foo(Middle a1)
Sub:   foo(Upper a1)
```

```
Calls on Super object:
1    Super.foo(Lower@765291)

Calls on Sub object in Super reference:
3    Super.foo(Lower@765291)

Calls on Sub object in Sub reference:
5    Super.foo(Lower@765291)
6    Sub.foo(Upper@26e431)
```

New in Java 1.5 – Generics

- Defining a generic class:

```
public interface List<E> {
   void add(E x);
   Iterator<E> iterator();
}
```

- Use:

```
List<String> words =
        new ArrayList<String>();
```

C++

- ISO/IEC 14882 Standard from 1998
- Originally developed by Bjarne Stroustrup

"C++ is a general purpose programming language with a bias towards systems programming that

- is a better C
- supports data abstraction
- supports object-oriented programming
- supports generic programming."

Bjarne Stroustrup

Hello World in C++

```
#include <iostream>

int main() {
    std::cout << "Hello, world!\n";
}
```

Some Features of C++

- Everything that C has
- Adds OO features
- Operator overloading
- Flexible template mechanism
- Large Standard Template Library STL
- Huge library support, though not uniform

Inheritance and Method Binding in C++

- Multiple Inheritance
- Change of visibility of inherited classes
- Static method binding by default
- Dynamic method binding with `virtual` keyword
- Abstract classes by using pure virtual functions
- Method overriding and overloading

Example for C++

```
class Super {
   public:
      virtual void m() = 0;
}

class Sub : public Super, private Impl,
  public IFace {
   public:
      void m();
}
void Sub::m() { cout << “Sub::m\n”; }
```

C++ References

`http://www.research.att.com/~bs/`

`http://www.open-std.org/jtc1/sc22/wg21/`

C#

"C# (pronounced "C Sharp") is a simple, modern, object oriented, and type-safe programming language.
It will immediately be familiar to C and C++ programmers.
C# combines the high productivity of Rapid Application Development (RAD) languages and the raw power of C++."

ECMA-334 C# Language Specification

Hello World in C#

```
using System;

class Hello {
    static void Main() {
      Console.WriteLine("hello, world");
    }
}
```

Some Features of C#

Basically everything from Java, plus:

- Definable value types `struct` and `enum`
- Four kinds of parameters: value, reference (`ref` keyword), and output parameters (`out` keyword), and parameter arrays (`params` keyword)
- Properties
- Indexers
- Operator overloading
- Delegates & Events
- Unsafe code sections

Array types: Rectangular or Jagged

```
class Test {
 static void Main() {
    int[] a1;   // single-dim array of int
    int[,] a2;  // 2-dimensional array of int
    int[,,] a3; // 3-dimensional array of int
    int[][] j2; // "jagged" array: array of
                // (array of int)
    int[][][] j3; // array of (array of
                  // (array of int))
  }
}
```

Inheritance and Method Binding in C#

- Single implementation inheritance
- Multiple interface subtyping
- Static method binding by default
- Dynamic method binding with `virtual` keyword
- Abstract classes and methods by using `abstract` keyword
- Keywords `override` and `new` distinguish between method overriding or hiding

Example for C#

```
using System;

public abstract class Super {
   public abstract void m();
}
public class Sub : Super, IFace {
   public [override | new] void m() {
      Console.WriteLine(“Sub.m”);
   }
}
```

C# References

http://msdn.microsoft.com/library/
default.asp?url=/library/en-us/cscon/
html/vcoricstartpage.asp

(base URL) + vclrfaquicksurveyof
(one word) csharpfeatures_pg.asp

http://genamics.com/developer/
csharp_comparative.htm

http://www.go-mono.com/

Questions?