Konzepte objektorientierter Programmierung

Prof. Dr. Peter Müller

Werner Dietl

Software Component Technology

Exercises 6: Information Hiding

Wintersemester 03/04

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Homework – Exercise 1

```
class C1 {
  private void m() {
     System.out.println("C1.m"); }

  protected void l() {
     m(); }
}

public class Test4 extends C1 {
  private void m() {
     System.out.println("C2.m"); }

  public static void main( String[] args ) {
     C1 c1 = new Test4();
     c1.l();
  }
}
```

Homework – Exercise 1

```
class C1 {
  private void m() {
     System.out.println("C1.m"); }

  protected void l() {
     m(); }
}
```

Output:
`C1.m`

Objects of class `Test4` have 4 different methods!

```
public class Test4 extends C1 {
  private void m() {
     System.out.println("C2.m"); }

  public static void main( String[] args ) {
     C1 c1 = new Test4();
     c1.l();
  }
}
```

Homework – Exercise 2

```
class C1 {
  private void m() {
     System.out.println("C1.m"); }
  protected void l() { m(); }
}
public class Test5 extends C1 {
  private void m() {
     System.out.println("C2.m"); }
  protected void l() { m(); }

  public static void main( String[] args ) {
     C1 c1 = new Test5();
     c1.l();
  }
}
```

Homework – Exercise 2

```
class C1 {
  private void m() {
     System.out.println("C1.m"); }
  protected void l() { m(); }
}
public class Test5 extends C1 {
  private void m() {
     System.out.println("C2.m"); }
  protected void l() { m(); }

  public static void main( String[] args ) {
     C1 c1 = new Test5();
     c1.l();
  }
}
```

Output:
`C2.m`

Objects of class `Test5` have 5 different methods!

Access Modifiers

- Following code in one package

Private Access

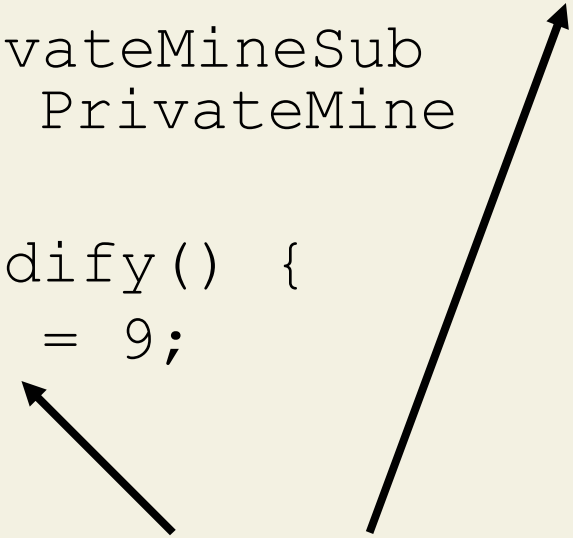
```
class PrivateMine {
  private int value;
}

class PrivateMineSub
  extends PrivateMine
{
  void modify() {
    value = 9;
  }
}
```

```
PrivateMineSub a =
 new PrivateMineSub();

a.value = 6;
a.modify();

System.out.println(
  "a.value = " +
     a.value );
```



Compilation Error!

Default Access

```
class DefaultMine {
    int value;
}

class DefaultMineSub
  extends DefaultMine
{
    void modify() {
        value = 9;
    }
}
```

```
DefaultMineSub a =
 new DefaultMineSub();

a.value = 6;
a.modify();

System.out.println(
  "a.value = " +
     a.value );
```

Protected Access

```
class ProtectedMine {
  protected int value;
}

class ProtectedMineSub
  extends
  ProtectedMine {
    void modify() {
        value = 9;
    }
}
```

```
ProtectedMineSub a =
  new
  ProtectedMineSub();

a.value = 6;
a.modify();

System.out.println(
  "a.value = " +
     a.value);
```

Access Modifiers

- Following code in two packages p1 and p2

Package p1

```
public class
DefaultMine
{
  int value;

  public int
  getValue() {
    return value;
  }
}
```

```
public class
ProtectedMine
{
  protected int value;

  public int
  getValue() {
    return value;
  }
}
```

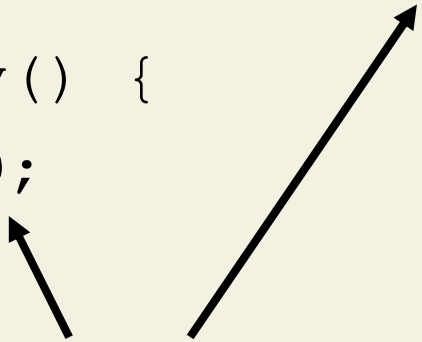
Package p2

```
public class
DefaultMineSub
extends p1.DefaultMine
{
  void modify() {
    value = 9;
  }
}
```

```
DefaultMineSub a =
 new DefaultMineSub();

a.value = 6;
a.modify();

System.out.println(
   "a.value = " +
      a.getValue());
```



Compilation Error!

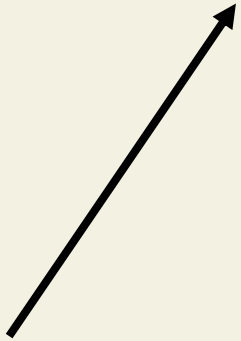
Package p2

```
public class
ProtectedMineSub
extends
  p1.ProtectedMine
{
  void modify() {
    value = 9;
  }
}
```

```
ProtectedMineSub a =
 new
   ProtectedMineSub();

a.value = 6;
a.modify();

System.out.println(
  "a.value = " +
     a.getValue());
```



Compilation Error!

Inner Classes – LinkedList Example

```
public class LinkedList {
   private static class Node {
      private Node next;
      private Object data;

      private Node(Object o, Node n) {
         data = o;
         next = n;
      }
   }
   private Node first;
```

LinkedList Implementation

```
LinkedList() { first = null; }

public void push( Object o ) {
    Node n = new Node(o, first);
    first = n;
}

public Object pop() {
    Object res = first.data;
    first = first.next;
    return res;
}

public Node getFirst() { return first; }
```

LinkedList Test Program

```
public static void main(String[] args)
{
    LinkedList l = new LinkedList();

    l.push( "Hello, World!" );
    l.push( "Oh, my gosh!" );

    System.out.println( l.pop() );
    System.out.println( l.pop() );
}
```

In a different class we have:

```
LinkedList l = new LinkedList();
l.push("Hello!");

Object o = l.getFirst();

if( o instanceof LinkedList.Node ) {
  System.out.println("Ha!");
}

System.out.println("First: " + o);
```

Compilation Error!

More on static inner classes

```
public class MyClassIsMyCastle {
  private static int streetno = 159;

  private static class FirstFloor {
    private static class DiningRoom {
      private static int size = 36;

      private static void myMessage() {
        System.out.println("Street No: " +
                           streetno);

      }
    }
  }
```

From: Poetzsch-Heffter 2000, Page 89

```
...
  private static class SecondFloor {
    private static int size = 16;

    private static void myMessage() {
      System.out.println("Dining-Room size: "+
           FirstFloor.DiningRoom.size );
    }
  }

  public static void main( String[] args ) {
    FirstFloor.DiningRoom.myMessage();
    SecondFloor.myMessage();
  }
}
```

Non-static Inner Classes

```
public class LinkedList {
  class ListIterator {
    private Node next = first;
    Object next() { …
      Object elem = next.data;
      next = next.next; return elem;
    }
  }

  ListIterator getIterator() {
    return new ListIterator(); }
  …
```

Usage of the ListIterator

```
LinkedList l = new LinkedList();

l.push( "Hello, World!" );
l.push( "Oh, my gosh!" );

LinkedList.ListIterator it = l.getIterator();

Object el;
while( (el = it.next()) != null ) {
  System.out.println("Element: " + el);
}
```

Non-static Inner Classes

- Can access private instance variables of the surrounding class
- In case of name conflicts the instance of the surrounding class can be accessed by e.g.:

```
private Node next = LinkedList.this.first;
```

- Alternative to instantiate an inner class:

```
LinkedList.ListIterator it =
                    l.new ListIterator();
```

- [To implement access to private fields the Java compiler actually creates some default-access methods!]