Konzepte objektorientierter Programmierung

Prof. Dr. Peter Müller

Werner Dietl

Software Component Technology

Exercises 6: Information Hiding

Wintersemester 05/06

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(); }
}

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();
  }
}
```

Output:
`C1.m`

Objects of class `Test4` have 4 different methods!

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!

Homework – Exercise 3 a

```
public class AList<T> {
  private T[] tarray;
  private int nextidx;

  public AList() {
     // the cast to T[] is unchecked!
     tarray = (T[]) new Object[10];
     nextidx = 0;    }

  public AList( T[] d ) {
     tarray = d;
     nextidx = d.length;    }
```

Homework – Exercise 3 b

- getFirstNode has protected visibility
- All classes in this package can call the method
- All subclasses can call the method

Homework – Exercise 3 c

The second constructor

```
public AList( T[] d ) {
   tarray = d;
   nextidx = d.length;   }
```

captures the array that is given as parameter.

The modification of the `ia` array is observable!

The output of the main program changes!

Homework – Exercise 4 a

```
public class FourA {
    private int a;
    private int b;
    /*@ invariant a >= b; @*/
    /*@ requires ta >= 0; @*/
    public FourA(int ta) {
      a = ta;
      b = 0;
    }
    public void increment() {
      ++a;
      ++b;
    }
}
```

OK!

Homework – Exercise 4 b

```
public class FourB {
    public int a;
    public int b;
    /*@ invariant a >= b; @*/
    /*@ requires ta >= 0; @*/
    public FourB(int ta) {
      a = ta;
      b = 0;
    }
    public void increment() {
      ++a;
      ++b;
    }
}
```

Not OK!

Homework – Exercise 4 c

```
public class FourC {
    private int a;
    private int b;
    public int c;
    /*@ invariant a >= b; @*/
    /*@ requires ta >= 0; @*/
    public FourC(int ta) {
      a = ta;
      b = 0;
    }
    public void increment() {
      ++a;
      ++b;
    }
}
```

OK!

Homework – Exercise 5

```
class Exa {
  /* invariant \type(a) <: A; */
  Object a;

  /* requires \type(c) <: C; */
  /* ensures \type(\result) <: B; */
  Object m(Object c) { ... }
}
```

Temporary violations of invariant possible!

Homework – Exercise 6

```
class Super {
    B m(C c) { ... }
}

class Sub extends Super {
    E m(F c) { ... }
}

C <: F
E <: B
```

Homework – Exercise 6

- Assume: F <: C, D <: C and B <: E

```
Super s = new Sub();
C c1 = new D();
B b1 = s.m( c1 );
```

Access Modifiers

- Following code in one package

Private Access

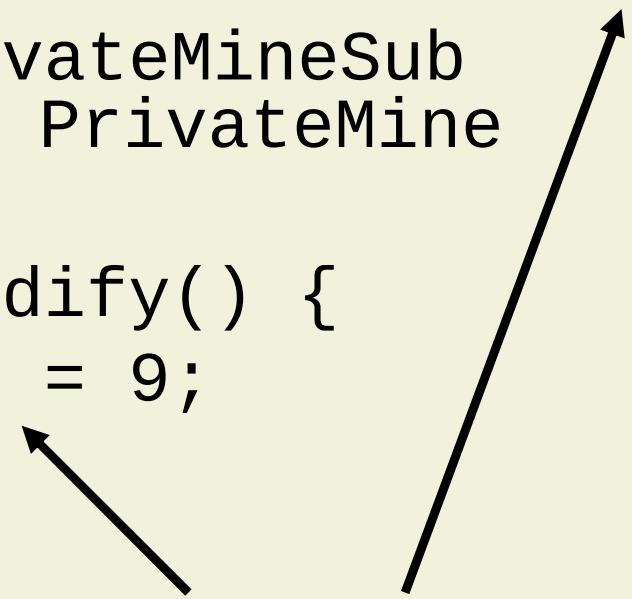
```
class PrivateMine {
  private int value;
}

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;
  }
}
```

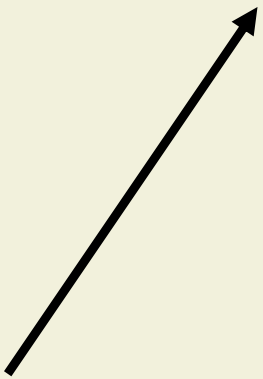
Package p2

```
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!

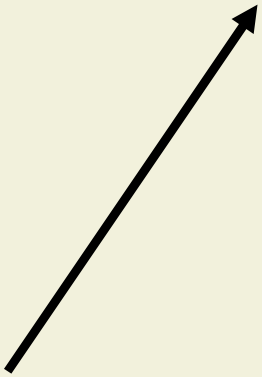
Package p2

```
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!

Questions?