Konzepte objektorientierter Programmierung

Prof. Dr. Peter Müller

Werner Dietl

Software Component Technology

Exercises 2: Java Overview

Wintersemester 03/04

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

2. Java Overview

- Quick introduction to Java
- Objects, Classes, Interfaces
- Focus on main elements for object-oriented programming
- Nearly no explanation of the standard libraries

Java, J2EE, J2SE, J2ME, and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

Classes and Objects – Basics

- Classes are the basic building block of Java
- Every attribute and method belongs to one class
- Every class inherits from the base class `Object`
- `Object` is the root of all classes, even if it is not mentioned as superclass
- `Object` provides basic features like string conversion, comparison, cloning, synchronization and meta information

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Why a class `Object` as root of the type tree?

- Create an array that can contain any object:

```
Object[] arr = new Object[10];
arr[0] = "A String here...";
arr[1] = new Thread();
```

- Methods that can take any object as argument:

```
public void log( Object message ) {
  System.err.println("Message: " + message);
}
```

- Ensure some standard methods, e.g. `toString` for String conversion and `equals` for equality

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Types of Arrays

```
void readArray( Object[] arr ) {
  System.out.println("Array[0]=" + arr[0]);
}

void writeArray( Object[] arr ) {
  arr[0] = "My array!"; // move other elems …
}                       java.lang.ArrayStoreException!

Integer[] ia = new Integer[2];
ia[0] = new Integer(5);
ia[1] = new Integer(6);

readArray( ia );
writeArray( ia );
```


ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Primitive Data Types

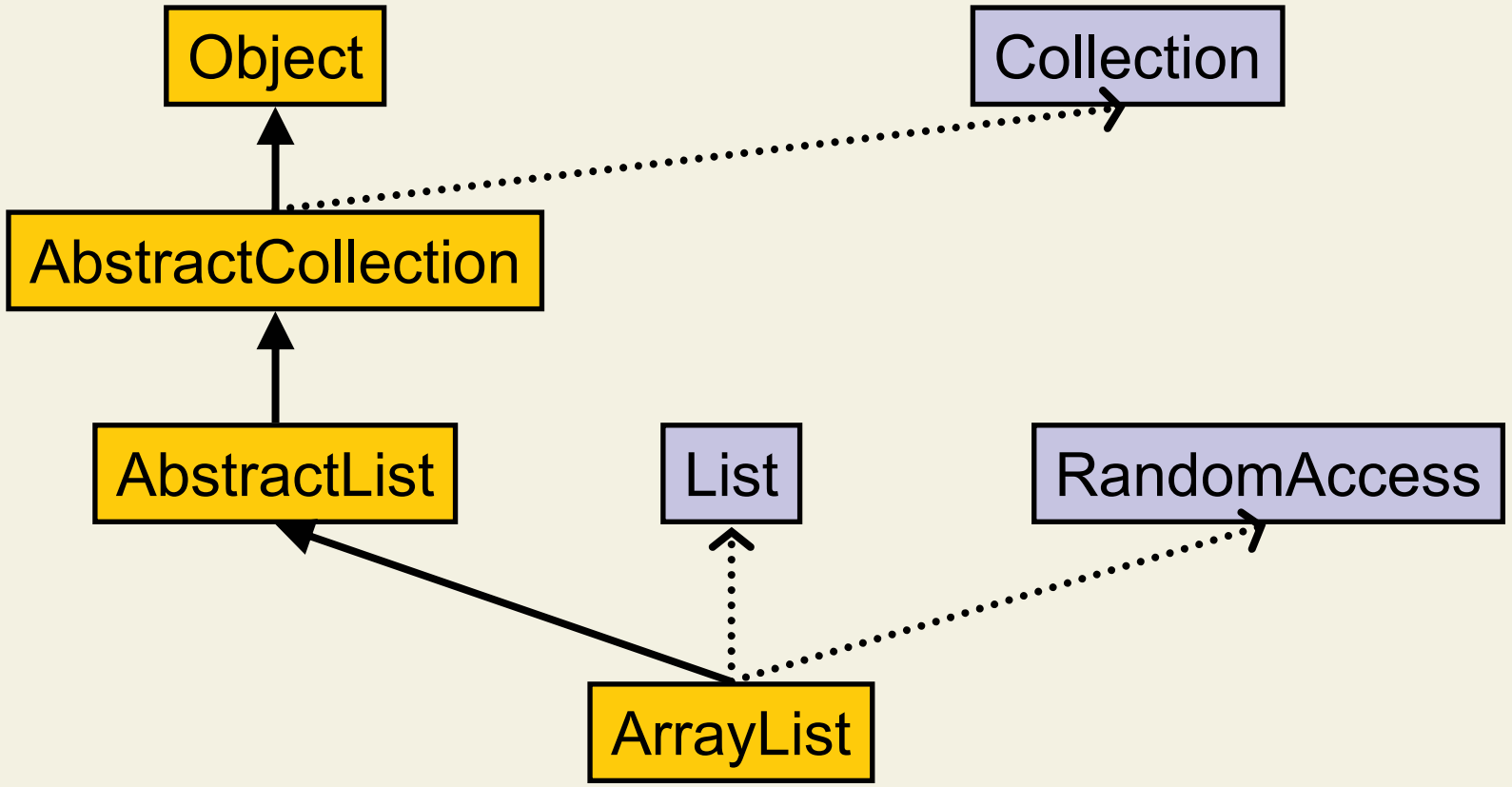
- These are: `boolean, byte, short, int, long, char, float, double`
- They are not subclasses of `Object` and can not be used where objects are required (e.g. Collections)
- In those cases the wrapper classes can be used, e.g. `Integer` or `Float`
- Precision and behavior of all types exactly defined
-
```
Object[] arr = new Object[10];
arr[0] = 10;
arr[1] = new Integer(10);
```
Compilation Error!
Ok.

Classes and Objects – Inheritance

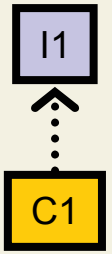
- Single Implementation Inheritance:
  - Only one other class can be extended
  - Methods and attributes of the superclass are inherited and can also be used on the subclass (if visible)
  - Subtyping and Inheritance at the same time
    → Subclassing
- Multiple Interface Subtyping:
  - Interfaces just specify which methods a class must have to comply with the interface
  - No implementation associated with the Interfaces

Inheritance Example

- Class java.util.ArrayList:



Quiz about Type Casts 1:

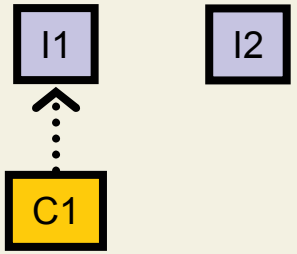



```
interface I1 {}
class C1 implements I1 {}

  public class Test1 {
    public static void main( String[] args ) {
      C1 c1 = new C1();
      I1 i1 = (I1) c1;
    }
  }
```

- Compile-time errors? No.
- Run-time errors? No.

Quiz about Type Casts 2:



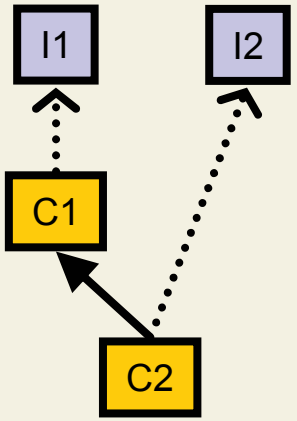

```
interface I1 {}
interface I2 {}
class C1 implements I1 {}

  public class Test2 {
    public static void main( String[] args ) {
      C1 c1 = new C1();
      I2 i2 = (I2) c1;
    }
  }
```

- Compile-time errors? No.
- Run-time errors? java.lang.ClassCastException!

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Quiz about Type Casts 3:




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

  public class Test3 {
    public static void main( String[] args ) {
      C1 c1 = new C2();
      I2 i2 = (I2) c1;
    }
  }
```

- Compile-time errors? No.
- Run-time errors? No.

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Overloading and Overriding

- The same method name can be used with different parameter types → Overloading
- The most specific method signature is selected for each method invocation
- Subclasses can define a different implementation of a method → Overriding
- For overriding all method parameter types have to stay the same, otherwise the method is overloaded

Overloading Example

```
static void func( String str, Object obj ) {
  System.out.println("String, Object");
}
static void func( Object obj, String str ) {
  System.out.println("Object, String");
}

public static void main( String[] args ) {
  func( "Name", new Integer(10) );
  func( new Integer(10), "Name" );
  func( "Name", "10" );
}
```

Compilation Error!

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

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();
```


ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

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)
```


ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

How to compile and run Java programs

- Java source code is saved in `.java` files
- Every public class must be put into its own .java file
- `javac` is used to compile Java files:

```
javac Test.java
```

- Compilation results in one or more `.class` files
- Set correct CLASSPATH environment variable
- One of your classes needs a special method:

```
public static void main( String[] args )
{ … }
```

- To execute the main method of class `Test` type:

```
java Test
```


ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Exception Handling

Basic syntax:

```
try {
    // dangerous code or …
    throw new MyException();
    …
} catch( MyException me ) {
    // handle my own exception
} catch( Exception e ) {
    // handle other exceptions
} finally {
    // do something that definitely has to happen
}
```


ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Exception Handling

- Catch-Blocks contain the code that handles the exceptional situation; the first catch-block whose type is a supertype of the current exception will be used → order the exceptions from most specific to least specific
- Finally-Block contains code that needs to be executed after the try-block, with or without an exception occurring
- Exceptions that are not caught need to be declared in the method signature! Example:

```
void m() throws MyException { … }
```


ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Assertions (since Java 1.4)

- Two versions of assertions:
```
assert Expression1 ;
assert Expression1 : Expression2 ;
```
- Java evaluates `Expression1` and if it is false throws an `AssertionError` with the value of `Expression2` as detail message, if present.
- Compile like this:
```
javac -source 1.4 MyClass.java
```
- At runtime enable/disable assertions selectively per package or class:
```
java –enableassertions:pack.age... Test
```
- `http://java.sun.com/j2se/1.4.2/docs/guide/lang/assert.html`

Assertions for Pre- and Post-Conditions

- Assertion statements can be used to simulate Design-by-Contract in a simple way
- For public methods use if-statements to check the input, because assertions can be turned off
- But for internal methods you can use asserts to make sure the preconditions are met
- Assertions at the end of a method can be used to establish postconditions
- Consistently adding checks to the postconditions can establish a class invariant

Assertions Example

- Internal function for updating an attribute:

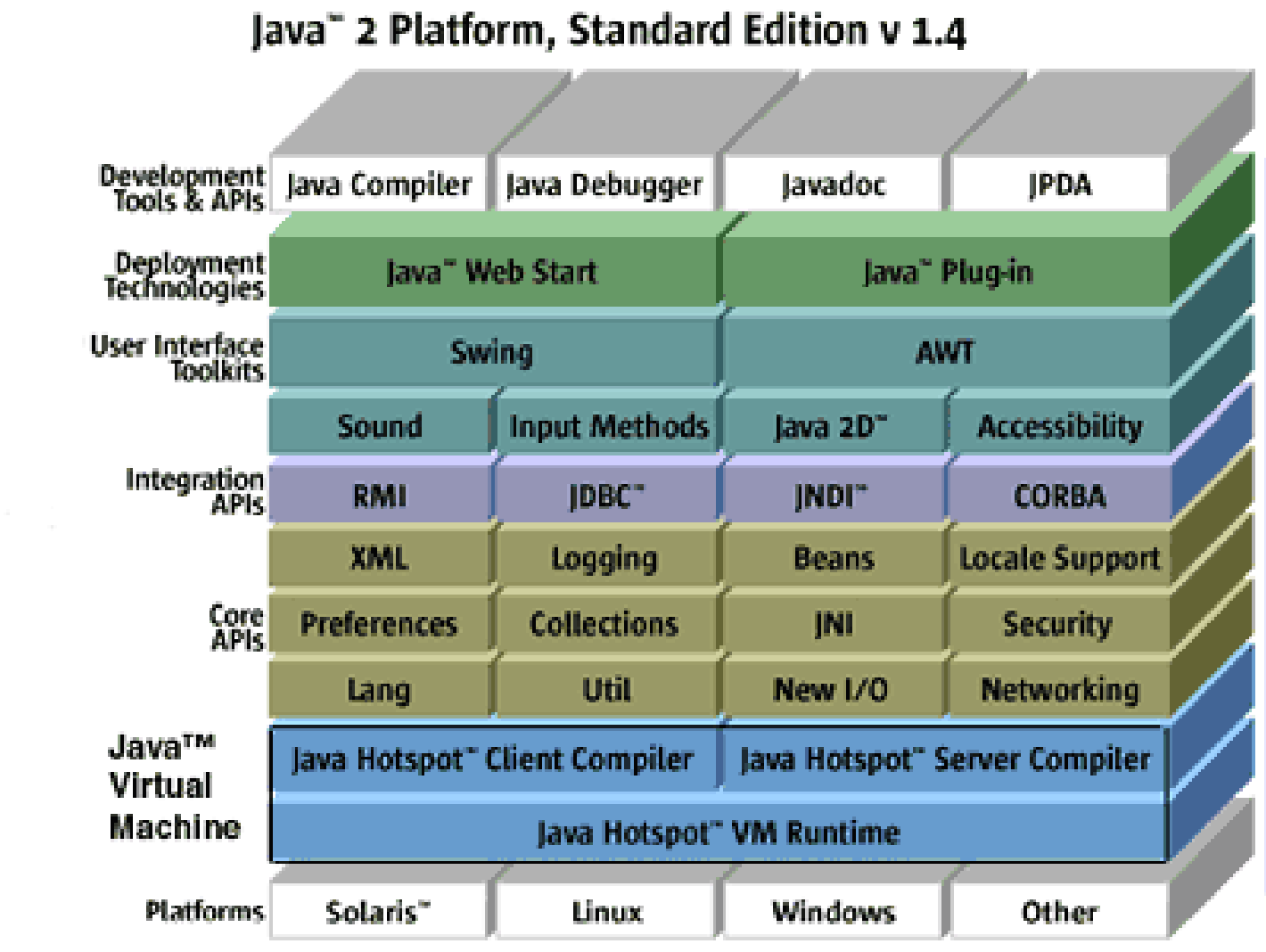
```
private int attr;
void setAttr( int newAttr ) {
    assert newAttr > 0 :
           "New attribute value <= 0!";
    attr = newAttr;
}
```

- An invalid call at runtime results in:

```
Exception in thread "main" java.lang.AssertionError:
           New attribute value <= 0!
    at Test9.setAttr(Test9.java:8)
    at Test9.main(Test9.java:23)
```

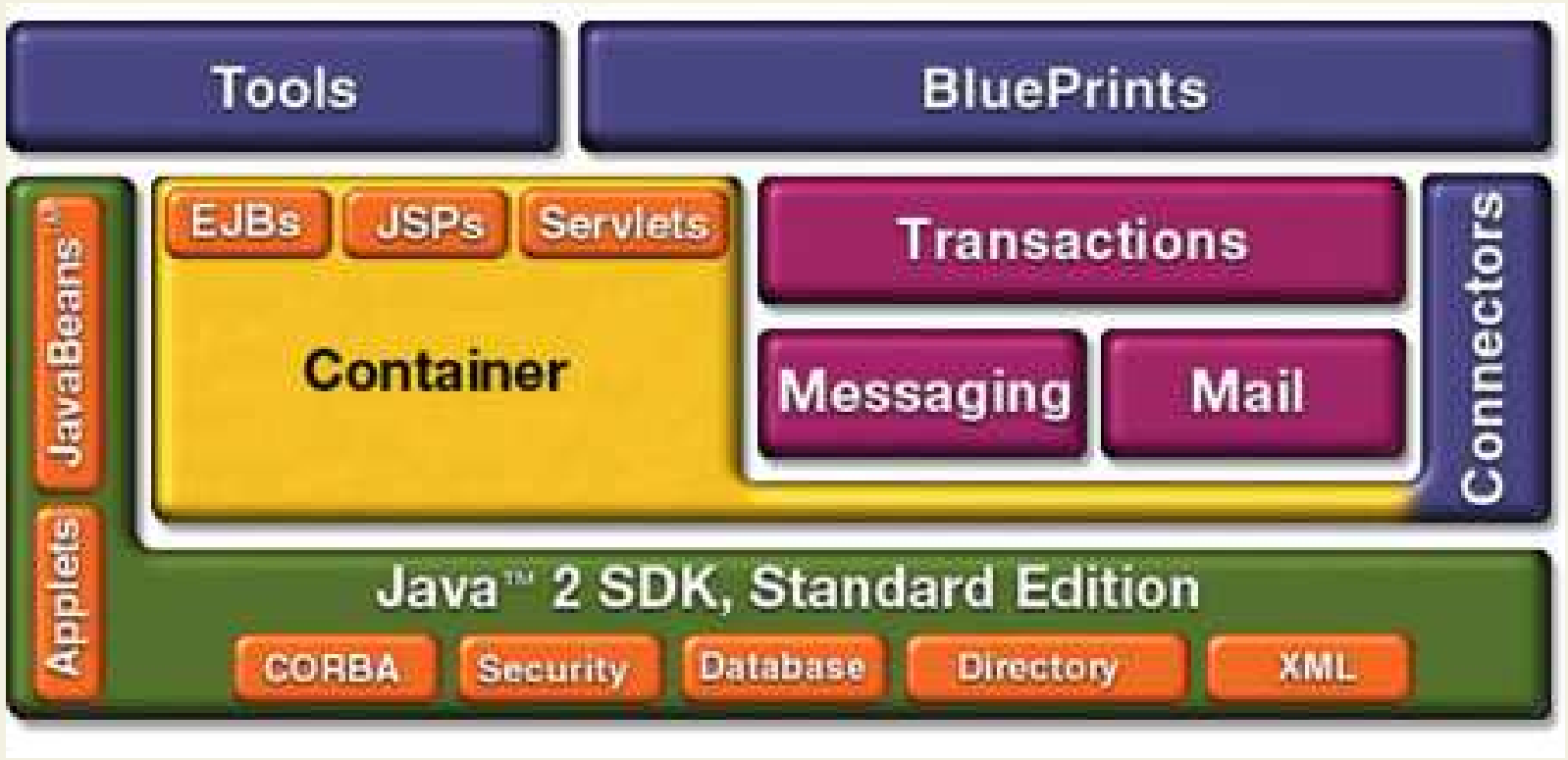

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Java 2 Platform, Standard Edition



ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Java 2 Platform, Enterprise Edition



ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Other Java Technologies

- Java Access Bridge
- Java Advanced Imaging
- Java Authentication and Authorization Service (JAAS)
- Java Communications API (JCA)
- Java Cryptography Extension (JCE)
- Java Data Objects (JDO)
- JavaMail API
- Java Management Extensions (JMX)
- Java Media Framework (JMF)
- Java Naming and Directory Interface (JNDI)
- Java Secure Socket Extension (JSSE)
- Java Speech API
- Java 3D API

What's to come in Java 1.5

- Generics:

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

- Enhanced `for`-loop:

```
void cancelAll(Collection c){
   for (Object o : c)
        ((TimerTask)o).cancel();  }
```


ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

What's to come in Java 1.5

- Autoboxing/unboxing

```
public class Freq {
   public static void main(String args[]) {
      Map<String, Integer> m =
         new TreeMap<String, Integer>();
      for(String word: args)
         m.put(word, m.get(word) + 1);
      System.out.println(m);
   }
}
```

- Other things: typesafe enums, static import, metadata, …

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Questions?

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich