Föreläsning 16

Andreina Francisco
(based on slides by Tobias Wrigstad)

Imperativ och objektorienterad programmering
Who uses Java?

[TIOBE Index for Java chart]

Source: www.tiobe.com

Highest Position (since 2001): #1 in Oct 2018

Lowest Position (since 2001): #2 in Mar 2015
## Very Long Term History

To see the bigger picture, please find below the positions of the top 10 programming languages of many years back. Please note that these are average positions for a period of 12 months.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Java</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>15</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>C++</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>C#</td>
<td>4</td>
<td>5</td>
<td>7</td>
<td>11</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Python</td>
<td>5</td>
<td>7</td>
<td>6</td>
<td>12</td>
<td>27</td>
<td>16</td>
<td>-</td>
</tr>
<tr>
<td>Visual Basic .NET</td>
<td>6</td>
<td>14</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>JavaScript</td>
<td>7</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>20</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PHP</td>
<td>8</td>
<td>6</td>
<td>4</td>
<td>5</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Perl</td>
<td>9</td>
<td>8</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>8</td>
<td>-</td>
</tr>
<tr>
<td>Delphi/Object Pascal</td>
<td>10</td>
<td>11</td>
<td>11</td>
<td>8</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Lisp</td>
<td>31</td>
<td>12</td>
<td>15</td>
<td>13</td>
<td>8</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Prolog</td>
<td>32</td>
<td>30</td>
<td>26</td>
<td>15</td>
<td>17</td>
<td>13</td>
<td>3</td>
</tr>
</tbody>
</table>

Källa: Tiobe
Java is Platform Independent

- Data types have a standard size
- All classes in the standard library are available on all machines
- Java's memory model is the same on all machines
- Your program will behave similarly on your friends computer
  
  You don't even have to recompile the program! — you can move it at once

  Caveat: Interface programs and OS-specific services

  C:\foo\bar.txt vs. /foo/bar.txt
### Java is Platform Independent

<table>
<thead>
<tr>
<th>Type</th>
<th>Default</th>
<th>Size</th>
<th>Example Literals</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>false</td>
<td>1 bit</td>
<td>true, false</td>
</tr>
<tr>
<td>byte</td>
<td>0</td>
<td>8 bits</td>
<td>(none)</td>
</tr>
<tr>
<td>char</td>
<td>\u0000</td>
<td>16 bits</td>
<td>'a', \u0041,</td>
</tr>
<tr>
<td>short</td>
<td>0</td>
<td>16 bits</td>
<td>(none)</td>
</tr>
<tr>
<td>int</td>
<td>0</td>
<td>32 bits</td>
<td>-2, -1, 0, 1, 2</td>
</tr>
<tr>
<td>long</td>
<td>0</td>
<td>64 bits</td>
<td>-2L, -1L, 0L, 1L, 2L</td>
</tr>
<tr>
<td>float</td>
<td>0.0</td>
<td>32 bits</td>
<td>1.23e100f,</td>
</tr>
<tr>
<td>double</td>
<td>0.0</td>
<td>64 bits</td>
<td>1.23456e300d,</td>
</tr>
</tbody>
</table>

Datatyperna är samma oavsett plattform
Automatic Memory Management

Objects know their size (but we may not!)

```java
new LinkedList();
```

Unreachable objects are reclaimed

```java
/// This program does not leak
LinkedList list = new LinkedList();
for (int i = 0; i < 1000000; ++i) {
    list.add(new Object());
}
list = null;
```

Unused objects are not...

```java
/// This program might "leak"
LinkedList list = new LinkedList();
for (int i = 0; i < 1000000; ++i) {
    list.add(new Object());
}
```
Metadata

- An object knows its origin

  ```java
  Object o = new Person();
  o instanceof Person // true
  Class c = o.getClass();
  c.newInstance(); // create a new person
  ```

- Reflection and introspection

  ```java
  Method m = o.getClass().getMethod("setName", String.class);
  m.invoke(o, "Barbara")
  for (Method m : c.getMethods()) { if (m.startsWith("test")) m.invoke(); }
  ```

- ...and more, e.g., array.length
Encapsulation

• Name-based encapsulation controls who can use/name a particular name

• Node is just a valid type inside LinkedList

• Requires active stance from you!

• The default (no access modifier) is “package” — i.e. accessible everywhere from the module

```java
public class Pair {
    private Object fst;
    private Object snd;
    Object getFst() { return this.fst; }
    void setFst(Object o) { this.fst = o; }
}
```

```java
public class LinkedList {
    private Node first = new Node();
    private class Node {
        Node next;
        Object element;
        public Node(Object o, Node n) {
            this.element = o;
            this.next = n;
        }
        public void prepend(Object o) {
            this.first =
                new Node(o, this.first);
        }
    }
}
```
The World's Richest Standard Library (?)

- Search for “java 10 api Class Name”
- Generated with Javadoc based on comments in the source code — (inspiration for D9)
- Included in packages. Most important packages for you:
  
  ```
  java.lang
  
  Basic objects, and system objects
  
  java.util
  
  Common data structures, StringTokenizer
  
  java.io
  
  I/O
  ```

http://docs.oracle.com/javase/X/docs/api/
java.util

Class HashMap<K,V>

java.lang.Object
  java.util.AbstractMap<K,V>
    java.util.HashMap<K,V>

Type Parameters:
  K - the type of keys maintained by this map
  V - the type of mapped values

All Implemented Interfaces:
  Serializable, Cloneable, Map<K,V>

Direct Known Subclasses:
  LinkedHashMap, PrinterStateReasons

public class HashMap<K,V> 
extends AbstractMap<K,V> 
implements Map<K,V>, Cloneable, Serializable

Hash table based implementation of the Map interface. This implementation provides all of the optional map operations, and permits null values and the null key. (The HashMap class is roughly equivalent to Hashtable, except that it is unsynchronized and permits nulls.) This class makes no guarantees as to the order of the map; in particular, it does not guarantee that the order will remain constant over time.
Constructor Summary

Constructors

<table>
<thead>
<tr>
<th>Constructor and Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>HashMap()</code></td>
</tr>
<tr>
<td>Constructs an empty <code>HashMap</code> with the default initial capacity (16) and the default load factor (0.75).</td>
</tr>
<tr>
<td><code>HashMap(int initialCapacity)</code></td>
</tr>
<tr>
<td>Constructs an empty <code>HashMap</code> with the specified initial capacity and the default load factor (0.75).</td>
</tr>
<tr>
<td><code>HashMap(int initialCapacity, float loadFactor)</code></td>
</tr>
<tr>
<td>Constructs an empty <code>HashMap</code> with the specified initial capacity and load factor.</td>
</tr>
<tr>
<td><code>HashMap(Map&lt;? extends K, ? extends V&gt; m)</code></td>
</tr>
<tr>
<td>Constructs a new <code>HashMap</code> with the same mappings as the specified <code>Map</code>.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Modifier and Type</th>
<th>Method and Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void</code></td>
<td><code>clear()</code></td>
</tr>
<tr>
<td></td>
<td>Removes all of the mappings from this map.</td>
</tr>
<tr>
<td><code>Object</code></td>
<td><code>clone()</code></td>
</tr>
<tr>
<td></td>
<td>Returns a shallow copy of this <code>HashMap</code> instance: the keys and values themselves are not cloned.</td>
</tr>
</tbody>
</table>
# Method Summary

## Methods

<table>
<thead>
<tr>
<th>Modifier and Type</th>
<th>Method and Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>void</strong></td>
<td><code>clear()</code>&lt;br&gt;Removes all of the mappings from this map.</td>
</tr>
<tr>
<td><strong>Object</strong></td>
<td><code>clone()</code>&lt;br&gt;Returns a shallow copy of this <code>HashMap</code> instance: the keys and values themselves are not cloned.</td>
</tr>
<tr>
<td><strong>boolean</strong></td>
<td><code>containsKey(Object key)</code>&lt;br&gt;Returns <code>true</code> if this map contains a mapping for the specified key.</td>
</tr>
<tr>
<td><strong>boolean</strong></td>
<td><code>containsValue(Object value)</code>&lt;br&gt;Returns <code>true</code> if this map maps one or more keys to the specified value.</td>
</tr>
<tr>
<td><strong>Set&lt;Map.Entry&lt;K,V&gt;&gt;</strong></td>
<td><code>entrySet()</code>&lt;br&gt;Returns a <code>Set</code> view of the mappings contained in this map.</td>
</tr>
<tr>
<td><strong>V</strong></td>
<td><code>get(Object key)</code>&lt;br&gt;Returns the value to which the specified key is mapped, or <code>null</code> if this map contains no mapping for the key.</td>
</tr>
<tr>
<td><strong>boolean</strong></td>
<td><code>isEmpty()</code>&lt;br&gt;Returns <code>true</code> if this map contains no key-value mappings.</td>
</tr>
<tr>
<td><strong>Set&lt;K&gt;</strong></td>
<td><code>keySet()</code>&lt;br&gt;Returns a <code>Set</code> view of the keys contained in this map.</td>
</tr>
<tr>
<td><strong>V</strong></td>
<td><code>put(K key, V value)</code>&lt;br&gt;Associates the specified value with the specified key.</td>
</tr>
</tbody>
</table>
**put**

```java
public V put(K key,
        V value)
```

Associates the specified value with the specified key in this map. If the map previously contained a mapping for the key, the old value is replaced.

**Specified by:**

```java
put in interface Map<K,V>
```

**Overrides:**

```java
put in class AbstractMap<K,V>
```

**Parameters:**

- key - key with which the specified value is to be associated
- value - value to be associated with the specified key

**Returns:**

the previous value associated with key, or null if there was no mapping for key. (A null return can also indicate that the map previously associated null with key.)

---

**putAll**

```java
public void putAll(Map<? extends K, ? extends V> m)
```

Copies all of the mappings from the specified map to this map. These mappings will replace any mappings that this map had for any of the keys currently in the specified map.

**Specified by:**

```java
```
Hundratals paket

Tusentals klasser
Strong Typing

- Java is strongly typed (C weakly!)
  
  We cannot treat one type as another
  
  Trying to do so will generate a clear runtime error
Parametric Polymorphism

```java
/// Revisiting previous examples — and improving them!
Person p1 = new Person();
Class<Person> cp = p1.getClass();
Person p2 = cp.newInstance();
```

```java
/// Revisiting previous examples — and improving them!
LinkedList<Person> list = new LinkedList<>(); //!!
list.add(new Object()); //!! Will not compile
Person p = list.first();
```

```java
/// Revisiting previous examples — and improving them!
public class Person implements Comparable<Person> {
    ...
}
```
No Segmentation Fault

Null Pointers in Java:

```java
Exception in thread "main" java.lang.NullPointerException
    at com.example.myproject.Book.getTitle(Book.java:16)
    at com.example.myproject.Author.getBookTitles(Author.java:25)
    at com.example.myproject.Bootstrap.main(Bootstrap.java:14)
```

Lookup in arrays with `index < 0` or `index >= array.length`

```java
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException
    at com.example.myproject.BookStore.getBook(BookStore.java:52)
    at com.example.myproject.BookStore.getLatest(BookStore.java:33)
    at com.example.myproject.Bootstrap.main(Bootstrap.java:17)
```

Note that the program prints which line it crashed on!!!!
```java
/// Bad type cast generates runtime error
BufferedReader in = null;
try {
    in = new BufferedReader(new FileReader("foo.in"));
    while (true) {
        node = node.next;
    }
    ...
} catch (NullPointerException e) {
    /// Went too far in the list!
    e.printStackTrace(System.err);
} finally {
    if (in != null) {
        in.close();
    }
}
```
References

• No pointer arithmetic
• A reference cannot be created out of nothing
• No dangling pointers
• A pointer is an address — an integer — an offset from 0
• A reference is and handle, a token through which one can access an object
• Sometimes we say pointers, nevertheless, it is obvious they are references by context
• A null-pekare is NOT a reference, it is the absence of a reference!
Jshell

- From JDK 9, Java has finally gotten a REPL (Read-Eval-Print Loop)

- Play with Java in an interactive, live environment

```java
[writo649@trygger:6 ~]$ jshell
| Welcome to JShell -- Version 10.0.2
| For an introduction type: /help intro

jshell> 42 + 42
$1 ==> 84

jshell> public class Test { public Test(int i) { this.i = i; } int i; }
| Error:
| cannot find symbol
| symbol: class public
| public class Test { public Test(int i) { this.i = i; } int i; }
| ^-----^

jshell> public class Test { public Test(int i) { this.i = i; } int i; }
| Error:
| missing return statement
| public class Test { public Test(int i) { this.i = i; } int i; }
| ^-------------^

jshell> public class Test { public Test(int i) { this.i = i; } int i; }
| created class Test

jshell> Test t = new Test()
| Error:
| constructor Test in class Test cannot be applied to given types;
| required: int
| found: no arguments
| reason: actual and formal argument lists differ in length
| Test t = new Test();
| ^____^

jshell> Test t = new Test(42)
t ==> Test@5e3a8624
```
Object Oriented?

- Hopefully you will also appreciate object orientation…
The Stack in Java vs. The Stack in C

Ditt C-program
Operativsystem
Hårdvara

Ditt Java-program
Virtuell Maskin
Operativsystem
Hårdvara
Kompilera och köra ett Java-program

$ javac MyProg.java
   Creates one or more .class files, one of which is called MyProg.class

$ java MyProg
   Startar den virtuella maskinen och laddar in MyProg och kör
Compiling your Java-program

```
public class Person {
    public String name;
    public Person(String name) {
        assert name != null : "Name == null!";
        this.name = name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getName() {
        return this.name;
    }
    public String toString() {
        return "Person(" + this.name + ")";
    }
}
```

```
public class Person {
    public java.lang.String name;
    static final boolean $assertionsDisabled;
    public Person(java.lang.String);
    public void setName(java.lang.String);
    public java.lang.String getName();
    public java.lang.String toString();
    static {};
}
```
Compiled from "Person.java"

```java
public class Person {
    public java.lang.String name;

    static final boolean $assertionsDisabled;

    public Person(java.lang.String);

    Code:
    0: aload_0
    1: invokespecial #1
        // Method java/lang/Object."<init>":()V
    4: getstatic     #2
        // Field $assertionsDisabled:Z
    7: ifne         24
    10: aload_1
    11: ifnonnull  24
    14: new         #3
        // class java/lang/AssertionError
    17: dup
    18: ldc         #4
        // String Name must not be null!
    20: invokespecial #5
        // Method java/lang/AssertionError."<init>":(Ljava/lang/Object;)
    23: athrow
    24: aload_0
    25: aload_1
    26: putfield    #6
        // Field name:Ljava/lang/String;
    29: return

    public void setName(java.lang.String);

    Code:
```
public class Person {
    public String name;
    public Person(String name) {
        assert name != null : "Name == null!";
        this.name = name;
    }
    public String getName() {
        return this.name;
    }
    public String toString() {
        return "Person(" + this.name + ")";
    }
}

// class java/lang/StringBuilder

// Method java/lang/StringBuilder."<init>"():V
// String Person(
// Method java/lang/StringBuilder.append:(Ljava/lang/
// String;)Ljava/lang/StringBuilder;
// Field name:Ljava/lang/String;
// Method java/lang/StringBuilder.append:(Ljava/lang/
// String;)Ljava/lang/StringBuilder;
// String )
// Method java/lang/StringBuilder.append:(Ljava/lang/
// String;)Ljava/lang/StringBuilder;
// Method java/lang/StringBuilder.toString():Ljava/lang/String;
Automatic Garbage Collection

- Objective: to give the programmer the illusion that memory is infinite
- Method: identify the junk data and release it automatically
- Definition of garbage: data that cannot be accessed by the program (no references)
- More formally, the object O is garbage if there is no path in the memory graph from any root (the variable on the stack, global variables, etc.) to O
- Two basic ways to do automatic garbage collection:
  - Reference counting
  - Tracing
Reference Counting

- Basic idea: each item saves information on how many people point to it
- When this counter reaches 0 - remove the object
- Each time a reference is created / deleted, update the reference counter:

```c
void *p = malloc(2048); // refcount 1
void *x = p; // refcount 2
p = NULL; // refcount 1
x = NULL; // refcount 0, free(x)
```

- Problem:
  
  Cyclic structures (see next pages)

  Långlivat minne som manipuleras ofta kostar, fast vi aldrig tar bort det
Heap

Root set

vi tar bort denna
Heap

Root set

vi tar bort denna

Läckage!
Tracing GC: Mark-Sweep

• When memory runs out:
  
  1. Follow the roots and select all items that can be reached
  2. Iterate all items and release all that are not marked in 1.

• The next slide shows the mark in the ground phase (1.)
vi tar bort denna
I nästa mark-fas markerar vi med annan färg

- Om något är grönt fortfarande efter denna fas är det skräp och skall tas bort
Dessa två kan nu säkert tas bort
KEEP CALM AND LOVE PROGRAMMING