Beyond Eiffel

these slides contain advanced material and are optional
Beyond Eiffel

• *Eiffel* was used in the course to introduce you to programming

• The goal is not to learn programming *Eiffel*

• The goal is to
  – Understand programming
  – Learn the concepts of programming
  – Learn how to *programm well*
How to program well

• Understand fundamental concepts of programming
• Understand when and how to apply these concepts
• Write code with *readability* in mind
• Know the language you are using
• Experience
• More experience
Which language should you use?

• All programming languages have advantages and disadvantages
  – Ease of use
  – Performance characteristics (speed, memory)
  – Applicability to problem domain
  – Availability of libraries and supporting tools
  – Personal experience
  – Company expertise / existing codebase
  – ...

• Know the problem you want to solve
• Select the language accordingly
### TIOBE index top 10 languages December 2012 (sum up to 80%)

<table>
<thead>
<tr>
<th>Rank</th>
<th>Language</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>C</td>
<td>18.7%</td>
</tr>
<tr>
<td>2.</td>
<td>Java</td>
<td>17.6%</td>
</tr>
<tr>
<td>3.</td>
<td>Objective-C</td>
<td>11.1%</td>
</tr>
<tr>
<td>4.</td>
<td>C++</td>
<td>9.2%</td>
</tr>
<tr>
<td>5.</td>
<td>C#</td>
<td>5.5%</td>
</tr>
<tr>
<td>6.</td>
<td>PHP</td>
<td>5.5%</td>
</tr>
<tr>
<td>7.</td>
<td>(Visual) Basic</td>
<td>5.2%</td>
</tr>
<tr>
<td>8.</td>
<td>Python</td>
<td>3.8%</td>
</tr>
<tr>
<td>9.</td>
<td>Perl</td>
<td>2.2%</td>
</tr>
<tr>
<td>10.</td>
<td>Ruby</td>
<td>1.7%</td>
</tr>
</tbody>
</table>

### Paradigms

<table>
<thead>
<tr>
<th>Paradigm</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object-oriented</td>
<td>58.5%</td>
</tr>
<tr>
<td>Procedural</td>
<td>36.9%</td>
</tr>
<tr>
<td>Functional</td>
<td>3.2%</td>
</tr>
<tr>
<td>Logical</td>
<td>1.4%</td>
</tr>
</tbody>
</table>

### Type systems

<table>
<thead>
<tr>
<th>Type system</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Statically typed</td>
<td>71.4%</td>
</tr>
<tr>
<td>Dynamically typed</td>
<td>28.6%</td>
</tr>
</tbody>
</table>

Learning a new language consists of

- Learning the syntax (fast)
- Mapping known programming concepts to new syntax (fast)
- Learning the conventions (medium)
- Learning the libraries (long)
Some concepts in various languages

• Namespaces
• Encapsulation
• Inheritance
• Generics
• Contracts
• Function objects
Namespaces

• Global (Eiffel)
• Directory-based packages (Java)
  – Warnings if directory structure does not follow packages
• File-based modules (Python)
  – Module name = file name
• User-declared (C#)
  – Declare (multiple) arbitrary namespaces per file
Encapsulation

• Export status (Eiffel)
  – Granularity level of classes, no fully private
  – Attributes never writable from outside class

• Access modifier (Java, C#, C++, PHP)
  – Public (full access), private (only inside the class),
    protected (class + subclasses)

• Naming conventions (Python)
  – No access modifiers
  – Names starting with underscore should not be
    accessed from outside the class
Inheritance

- Static multiple inheritance (Eiffel, C++)
  - Name-Routine mapping defined at compile-time
  - Various conflict resolution schemes (renaming, virtual)
- Dynamic multiple inheritance (Python)
  - Inheritance ordering matters
  - Name resolution depth-first, left-to-right (+special cases)
- Single inheritance + Interfaces (Java, C#)
  - Single inheritance of full classes
  - Multiple inheritance of interfaces only
- Single inheritance (PHP)
Generics

- Generics (Eiffel)
- Generics (Java)
  - Safe co- and contravariance (Wildcards)
  - Type erasure
- Generics (C#)
  - No conformance
- Templates (C++)
- Dynamic typing (Python, PHP)
Contracts

• Built-in contracts (Eiffel)

• Contracts as a library (C#)
  – Library offering calls that are interpreted as preconditions / postconditions / invariants

• Assert statements (Java, C, Python)
  – Assertion in the beginning is a precondition
  – Assertion in the end is a postcondition
  – No contract inheritance
Function objects

• Agents (Eiffel)
  – Unique: open/closed arguments, open targets
• Function pointers (C)
• Functor (C++)
• Delegates (C#)
• Closures (Python)
• Anonymous inner classes (Java <8)


• Lambda expressions (Java 8)