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

Programming language frequency

TIOBE index top 10 languages December 2012 (sum up to 80%)

1.	C	18.7%
2.	Java	17.6%
3.	Objective-C	11.1%
4.	C++	9.2%
5.	C#	5.5%
6.	PHP	5.5%
7.	(Visual) Basic	5.2%
8.	Python	3.8%
9.	Perl	2.2%
10.	Ruby	1.7%

Paradigms

Object-oriented	58.5%
Procedural	36.9%
Functional	3.2%
Logical	1.4%

Type systems

Statically typed	71.4%
Dynamically typed	28.6%

Source: http://www.tiobe.com/index.php/tiobe_index

Learning a new language

- 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 acccess), 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)

See http://en.wikipedia.org/wiki/Function_object

- Lambda expressions (Java 8)
  - http://www.informit.com/articles/article.aspx?p=1963535&seqNum=2