ETH Zürich
Chair of Software Engineering

Java and C# in depth

Carlo A. Furia, Marco Piccioni, Bertrand Meyer

Object-oriented programming crash course (a.k.a. OO < 9 slides)

What's an OO programming language?

One with:

- ...

What's an OO programming language?

One with:

- Abstract Data Types
  - Classes (Types)
  - Objects (Instances)
- Encapsulation
  - information hiding, interfaces, ...
- Inheritance
- Polymorphism
- Dynamic binding/dispatching
- Genericity

Classes and Objects

Classes, Types and Objects

- A (non-generic) class defines a type
  - an abstraction represented as a set of features/members
    - attributes/fields/instance variables
    - routines/methods/member functions
  - serves as a mold for all its objects

- An object is an instance of a class
  - a chunk of memory shaped as its class dictates
  - can be attached to a reference of a certain type statically (resolved at compile time)…
  - … or dynamically (resolved at runtime)

Encapsulation

- Features can have different visibility/access levels
  - in particular, they can be accessible or inaccessible to clients of the class
  - E.g. public, private,…
- The set of features visible to a set of clients defines the interface for those clients

The two basic reuse mechanisms

- (Single) Inheritance relationship
  - a class gets to use the features of another class by inheriting from it
  - the derived class can also
    - introduce new features
    - redefine features of the parent class
  - good for modeling (really) long-lasting relationships

- Client relationship
  - a class gets to use the features of another class by declaring an attribute of that other class' type
  - good for modeling relationships more prone to change

Polymorphism and dynamic dispatching

- Dynamic type of an object can change through assignments
  - co-variant assignment: type of assigned is sub-type of type of receiver
  - contra-variant assignment: type of assigned is super-type of type of receiver
- Dynamic type can also change in generic parameters, method parameters, and return types
- Dynamic dispatching
  - dynamic binding of routine calls to routine bodies
  - the implementation is selected according to the dynamic type of a reference

Genericity (generics)

- Class definition can be generic with respect to other types
- A generic class doesn’t define a type but a family of types
- E.g. A list of strings and a list of integers (these are two types) share the same generic class List<T>
- Useful because the compiler disallows, for example, inserting an integer into a list of strings