Programming Paradigms

Data Abstraction and Object-Orientation (Part 1)

Prof. Dr. Michael Pradel

Software Lab, University of Stuttgart

Summer 2020

Data Abstraction

- **Goal: Describe class of memory objects and their associated behavior**
- **Abstract data type**
  - Set of values and set of operations
- **Example: Stack**
  - Values: Data on stack
  - Operations: push, pop, etc.

Classes

- **Most common form of data abstraction in today's PLs**
- **Two kinds of members**
  - Data members a.k.a. fields
  - Subroutine members a.k.a. methods
- **Class hides its implementation from clients**
- **Code reuse via inheritance**

Object-Oriented Programming

- **Objects**
  - Instances of classes (in class-based PLs, e.g., Java, C++)
  - Primary entities (in prototype-based PLs, e.g., Smalltalk, JavaScript)
- **(Most) data stored in fields of objects**
- **Objects call other objects' methods**

A Bit of History

Simula

- Developed in 1960s in Norway by Dahl and Nygaard
- First OO language
- Classes, objects, inheritance

Smalltalk

- Developed at Xerox PARC by Alan Kay and others
- Message-based programming, dynamic typing

C++, Eiffel, Ada95, Java, C#

Overview

- Encapsulation and Information Hiding 
- Inheritance
- Initialization and Finalization
- Dynamic Method Binding
- Mix-in and Multiple Inheritance

Encapsulation

- **Bundle related data with operations on the data**
  - Class members: Fields and methods
- **Instance-level vs. class-level**
  - Instance-level members: Specific to each individual object
  - Class-level members: Exist once for all objects of a class

Example

account.cpp

Information Hiding

- **Classes hide irrelevant details from their clients**
  - How the data is stored
  - How the behavior is implemented
- **Allows changing internals of a class without adapting the clients**

Getters and Setters

- **Hide details of how data is stored in fields**
- **Instead, access fields via**
  - Getter method: Returns the current value
  - Setter method: Set a new value
- **Clients read/write fields via getter/setter**

Properties/Accessors

- Special accessor methods for a field
- Transparent to clients: Looks like direct field access

```
// Example: C#
class Time {
  private double seconds;
  public double Hours {
    get { return seconds / 3600; }
    set {
      if (value < 0 || value > 24)
        // handle illegal argument
      seconds = value * 3600;
}}}
```

Properties/Accessors

- Special accessor methods for a field
- Transparent to clients: Looks like direct field access

```
// Example: C#
class Time {
  private double seconds;
  public double Hours {
    get { return seconds / 3600; }
    set {
      if (value < 0 || value > 24)
        // handle illegal argument
      seconds = value * 3600;
}}}
```

```
Time t = new Time();
t.Hours = 5;
double h = t.Hours;
```

Visibilities

- **Most class-based PLs provide visibilities for class members**
  - `private`: Visible only to the class itself
  - `public`: Visible to every client of the class
- **Expose members via `public` only when necessary**
  - Maximizes adaptability without affecting clients

Visibilities: PL Specifics

- **Java**
  - Default visibility: Visible in same package
  - `protected`: Visible in same package and all subclasses
- **C++**
  - `protected`: Visible in current class and subclasses
  - Friend classes: Can access private and protected members

Quiz: Encapsulation

Which of the following is true?

- Encapsulation bundles related data and operations, while hiding irrelevant details from clients.
- Clients of a class must adapt to how the class represents its internal data.
- `protected` means the same in Java and C++.
- Class-level members should always be public.

Please vote via Ilias.

Quiz: Encapsulation

Which of the following is true?

- Encapsulation bundles related data and operations, while hiding irrelevant details from clients.
- ~~Clients of a class must adapt to how the class represents its internal data.~~
- ~~`protected` means the same in Java and C++.~~
- ~~Class-level members should always be public.~~

Please vote via Ilias.