Summary

Object Oriented Programming

Piotr Fulmański

Summary

Summary

In short

The four pillars of object oriented programming are:

- **Inheritance**: child classes inherit data and behaviors from parent class;
- **Encapsulation**: containing information in an object, exposing only selected information;
- **Abstraction**: only exposing high level public methods for accessing an object;
- **Polymorphism**: many methods can do the same task.

Summary

Inheritance

Inheritance allows classes to inherit features of other classes. Organizes and **supports *polymorphism*** and ***encapsulation*** by enabling the definition and creation of specialized objects on the basis of more general ones. For specialized objects, it is not necessary to redefine the entire functionality, but only the one that is absent from a more general object. Typically, there are groups of objects called classes, and groups of classes called trees. They reflect common features of the objects. Inheritance **supports *reusability***.

Summary

Encapsulation

Encapsulation means containing all important information inside an object, and only exposing selected information to the outside world. Attributes and behaviors are defined by code inside the class template.

Then, when an object is instantiated from the class, the data and methods are encapsulated in that object. Encapsulation hides the internal software code implementation inside a class, and hides internal data of inside objects.

Ensures that an object cannot change the internal state of other objects in unexpected ways. Only the object's own methods are allowed to change its state. Each type of object presents its interface to other objects, which determines the acceptable methods of cooperation.

Summary

Encapsulation

Adds a layer of security, where the developer chooses what data can be seen on an object by exposing that data through public methods in the class definition.

Most programming languages have some ***access modifier***; the most typical are: *public*, *protected*, and *private* sections.

- **Public** is the limited selection of methods available to the outside world, or other classes within the program.

- **Protected** is only accessible to child classes.

- **Private** code can only be accessed from within that class.

All the data members should be made private to ensure the highest security of data. In special cases we can use public or protected access, but it is advised to keep the data members private always.

Summary

Encapsulation

The benefits of encapsulation are summarized here:

- **Adds security**: Only public methods and attributes are accessible from the outside.

- **Protects against common mistakes**: Only public fields and methods accessible, so developers don't accidentally change something dangerous.

- **Protects private informations**: Code is hidden in a class, only public methods are accessible by the outside developers.

- **Maintenance**: Most code undergoes updates and improvements.

- **Hides complexity**: No one can see what's behind the object's curtain.

- **Simplifies team work**: Makes it easier to collaborate with external developers.

Summary

Abstraction

Abstraction means that the user interacts with only selected attributes and methods of an object. Abstraction uses simplified, high level tools, to access a complex object. Abstraction is an extension of encapsulation.

Each object in the system serves as an abstract "performer" model that can do work, describe and change its state, and communicate with other objects in the system without revealing how everything is implemented.

Summary

Polymorphism

Polymorphism means **designing objects to share behaviors**. Using inheritance, objects can *override* shared parent behaviors, with specific child behaviors.
Polymorphism allows the same method to execute different behaviors in two ways: method *overriding* and method *overloading*. If this happens at runtime, it is called *late binding* or *dynamic binding*. If it can be determined during compilation, it is called *early binding* or *static binding*. Some languages provide more static (compile-time) solutions to polymorphism – for example, *templates* and *operator overloading*.

The benefits of polymorphism is that **objects of different types can be passed through the same interface**.

Criticism

Criticism

Luca Cardelli has claimed that OOP code is "intrinsically less efficient" than procedural code, that OOP can take longer to compile, and that OOP languages have "extremely poor modularity properties with respect to class extension and modification", and tend to be extremely complex.

According to Joe Armstrong, the principal inventor of Erlang, who is quoted as saying: *You wanted a banana but what you got was a gorilla holding the banana and the entire jungle*.

A study by Potok et al. has shown no significant difference in productivity between OOP and procedural approaches.

Lawrence Krubner in his article "*Object Oriented Programming is an expensive disaster which must end*" claimed that compared to other languages (LISP dialects, functional languages, etc.) OOP languages have no unique strengths, and inflict a heavy burden of unneeded complexity.

Wikipedia. Object-oriented programming. Criticism https://en.wikipedia.org/wiki/Object-oriented_programming#Criticism

Criticism

Paul Graham has suggested that OOP's popularity within large companies is due to "large (and frequently changing) groups of mediocre programmers". According to Graham, the discipline imposed by OOP prevents any one programmer from "doing too much damage".

Rob Pike, a programmer involved in the creation of UTF-8 and Go, has called object-oriented programming "the Roman numerals of computing" and has said that OOP languages frequently shift the focus from data structures and algorithms to types. Furthermore, he cites an instance of a Java professor whose "idiomatic" solution to a problem was to create six new classes, rather than to simply use a lookup table (Joseph Bergin, Russel WinderUnderstanding Object Oriented Programming, https://web.archive.org/web/20180829092402/http://csis.pace.edu/~bergin/patterns/ppoop.html in https://web.archive.org/web/20180814173134/http://plus.google.com/+RobPikeTheHuman/posts/hoJdanihKwb).

Wikipedia. Object-oriented programming. Criticism https://en.wikipedia.org/wiki/Object-oriented_programming#Criticism

Criticism

Object-oriented programming, whose essence is nothing more than programming using data with associated behaviors, is a powerful idea. It truly is. But it's not always the best idea.

Test

Test

1

Pick the term that relates to polymorphism:

A. Dynamic binding

B. Dynamic allocation

C. Static typing

D. Static allocation

Test

1

Pick the term that relates to polymorphism:

A. Dynamic binding

B. Dynamic allocation

C. Static typing

D. Static allocation

Test

2

An object that has more than one form is referred to as:

A. Inheritance

B. Interface

C. Abstract class

D. Polymorphism

Test

2

An object that has more than one form is referred to as:

A. Inheritance

B. Interface

C. Abstract class

D. Polymorphism

Test

3

The process by which one object can acquire the properties of another object:

A. Encapsulation

B. Inheritance

C. Polymorphism

Test

3

The process by which one object can acquire the properties of another object:

A. Encapsulation

B. Inheritance

C. Polymorphism

Test

4

When sub class declares a method that has the same type of arguments as a method declared by one of its superclasses, it is termed as:

A. Method overriding

B. Method overloading

C. Operator overloading

D. Operator overriding

Test

4

When sub class declares a method that has the same type of arguments as a method declared by one of its superclasses, it is termed as:

A. Method overriding

B. Method overloading

C. Operator overloading

D. Operator overriding

Test

5

Information hiding can also be termed as:

A. Data hiding

B. Encapsulation

C. Inheritance

Test

5

Information hiding can also be termed as:

A. Data hiding

B. Encapsulation

C. Inheritance

Test

6

Can objects of abstract classes be instantiated?

A. True

B. False

Test

6

Can objects of abstract classes be instantiated?

A. True

B. False

Test

7

The keyword which is used to access the method or member variables from the superclass:

A. super

B. using

C. is_a

D. has_a

Test

7

The keyword which is used to access the method or member variables from the superclass:

A. super

B. using

C. is_a

D. has_a

Test

8

Constructors are used to:

A. To build a user interface

B. Free memory

C. Initialize a newly created object

D. To create a sub-class

Test

8

Constructors are used to:

A. To build a user interface

B. Free memory

C. Initialize a newly created object

D. To create a sub-class

Test

9

The method with the same name or different return type and difference in the parameters either in number or type is known as:

A. Function overloading

B. Compile time overloading

Test

9

The method with the same name or different return type and difference in the parameters either in number or type is known as:

A. Function overloading

B. Compile time overloading

Test

10

What modifiers are allowed for methods in an Interface?

A. Abstract

B. Private

C. Public

Test

10

What modifiers are allowed for methods in an Interface?

A. Abstract

B. Private

C. Public

Test

11

Requires new question ;)

Test

11

Test

12

Which feature of OOP indicates code reusability?

A. Abstraction

B. Polymorphism

C. Encapsulation

D. Inheritance

Test

12

Which feature of OOP indicates code reusability?

A. Abstraction

B. Polymorphism

C. Encapsulation

D. Inheritance

Test

13

Which among the following doesn’t come under OOP concept?

A. Data hiding

B. Message passing

C. Platform independent

D. Data binding

Test

13

Which among the following doesn’t come under OOP concept?

A. Data hiding

B. Message passing

C. Platform independent

D. Data binding

Test

14

What is encapsulation in OOP?

A. It is a way of combining various data members and member functions that operate on those data members into a single unit

B. It is a way of combining various data members and member functions into a single unit which can operate on any data

C. It is a way of combining various data members into a single unit

D. It is a way of combining various member functions into a single unit

Test

14

What is encapsulation in OOP?

A. It is a way of combining various data members and member functions that operate on those data members into a single unit

B. It is a way of combining various data members and member functions into a single unit which can operate on any data

C. It is a way of combining various data members into a single unit

D. It is a way of combining various member functions into a single unit

Test

15

What is an abstraction in object-oriented programming?

A. Hiding the implementation and showing only the features

B. Hiding the important data

C. Hiding the implementation

D. Showing the important data

Test

15

What is an abstraction in object-oriented programming?

A. Hiding the implementation and showing only the features

B. Hiding the important data

C. Hiding the implementation

D. Showing the important data

Test

16

Which access specifier is usually used for data members of a class?

A. Protected

B. Private

C. Public

D. Default

Test

16

Which access specifier is usually used for data members of a class?

A. Protected

B. Private

C. Public

D. Default

Test

17

Which among the following best describes the Inheritance?

A. Using the data and functions into derived segment

B. Using already defined functions in a programming language

C. Using the code already written once

D. Copying the code already written

Test

17

Which among the following best describes the Inheritance?

A. Using the data and functions into derived segment

B. Using already defined functions in a programming language

C. Using the code already written once

D. Copying the code already written

Test

18

What happens if non static members are used in static member function?

A. Executes fine

B. Compile time error

C. Executes if that member function is not used

D. Runtime error

Test

18

What happens if non static members are used in static member function?

A. Executes fine

B. Compile time error

C. Executes if that member function is not used

D. Runtime error

Test

19

Which of the following best describes member function overriding?

A. Member functions having the same name in derived class only

B. Member functions having the same name and different signature inside main function

C. Member functions having the same name in base and derived classes

D. Member functions having the same name in base class only

Test

19

Which of the following best describes member function overriding?

A. Member functions having the same name in derived class only

B. Member functions having the same name and different signature inside main function

C. Member functions having the same name in base and derived classes

D. Member functions having the same name in base class only

Test

20

Which feature of OOP is exhibited by the function overriding?

A. Polymorphism

B. Encapsulation

C. Abstraction

D. Inheritance

Test

20

Which feature of OOP is exhibited by the function overriding?

A. Polymorphism

B. Encapsulation

C. Abstraction

D. Inheritance

Test

21

If data members are private, what can we do to access them from the class object?

A. Private data members can never be accessed from outside the class

B. Create public member functions to access those data members

C. Create private member functions to access those data members

D. Create protected member functions to access those data members

Test

21

If data members are private, what can we do to access them from the class object?

A. Private data members can never be accessed from outside the class

B. Create public member functions to access those data members

C. Create private member functions to access those data members

D. Create protected member functions to access those data members

Test

22

Which feature can be implemented using encapsulation?

A. Polymorphism

B. Overloading

C. Inheritance

D. Abstraction

Test

22

Which feature can be implemented using encapsulation?

A. Polymorphism

B. Overloading

C. Inheritance

D. Abstraction