Overview

- Understand Classes and Objects.

- Understand some of the key concepts/features in the Object Oriented paradigm.

- Benefits of Object Oriented Design paradigm.

Теми

- Обекти и класове Дефиниция на клас. Общи понятия и концепции.

- Модификатори на достъп в клас.

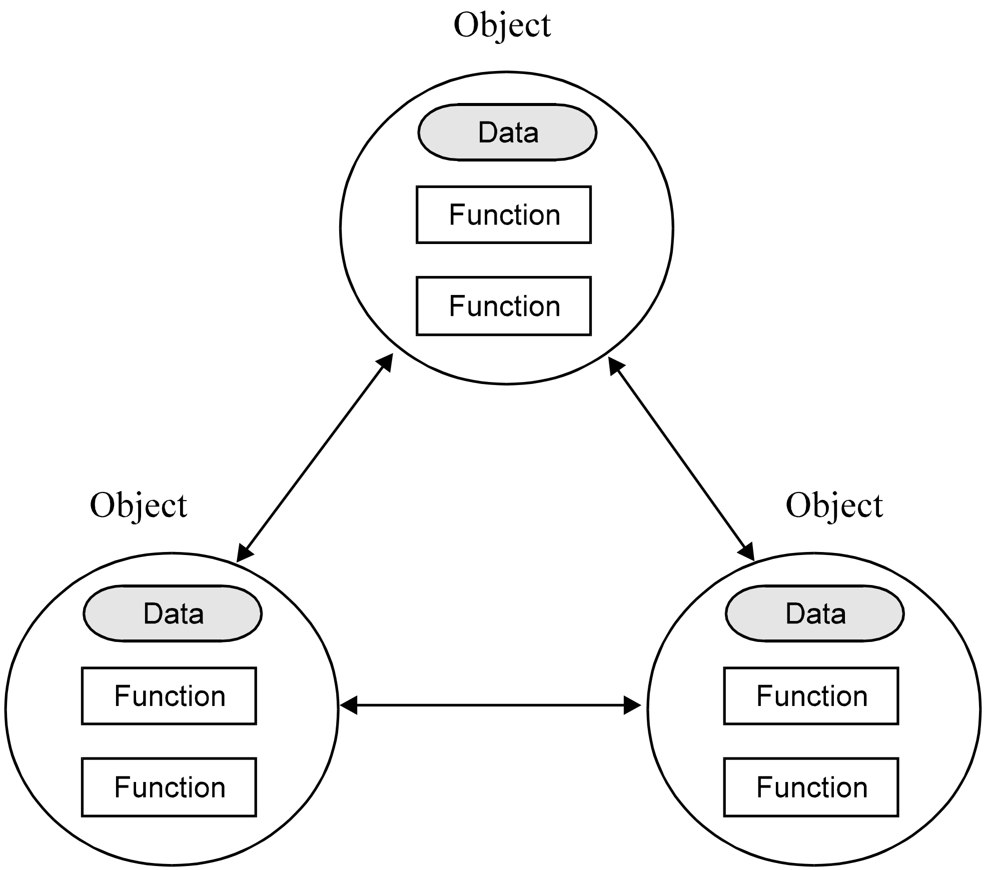
Object-Oriented Programming

Object-oriented programming is a method of implementation in which programs are organized as cooperative collections of objects, each of which represents an instance of some class, and whose classes are all members of a hierarchy of classes united via inheritance relationships.

Requirements

- It supports objects that are data abstractions with an interface of named operations and a hidden local state.
- Objects have an associated type [class].
- Types [classes] may inherit attributes from supertypes [superclasses].

OOP: model, map, reuse, extend



- Model the real world problem to user's perceive;
- Use similar metaphor in computational env.
- Construct reusable components;
- Create new components from existing ones.

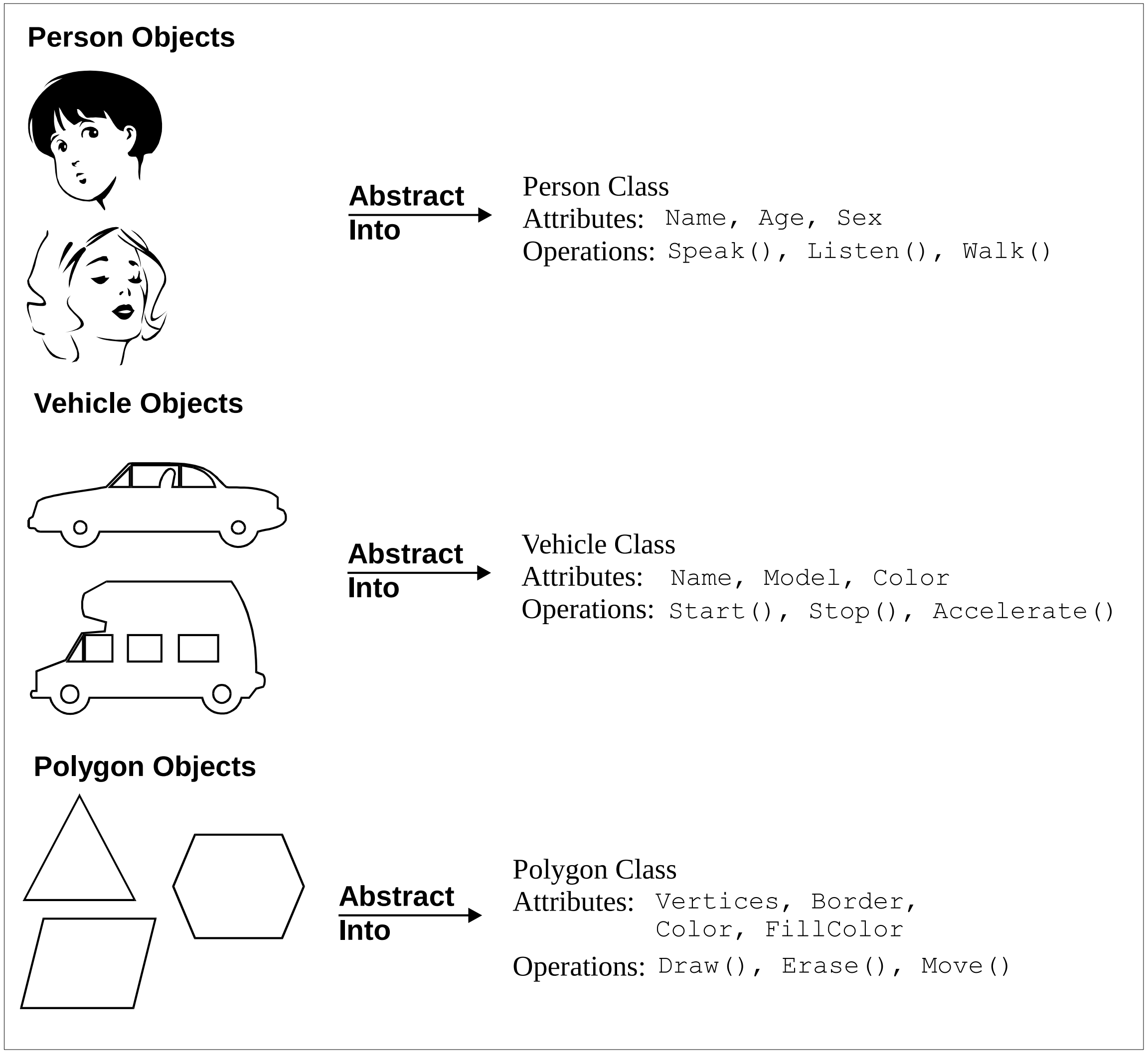
Class

- A class represents a template for several objects that have common properties.

- A class defines all the properties common to the object - *attributes* and *methods*.

- A class is sometimes called the object's type.

Classes: Objects with the same attributes and behavior



Person Objects

Abstract Into →

Person Class
Attributes: `Name, Age, Sex`
Operations: `Speak(), Listen(), Walk()`

Vehicle Objects

Abstract Into →

Vehicle Class
Attributes: `Name, Model, Color`
Operations: `Start(), Stop(), Accelerate()`

Polygon Objects

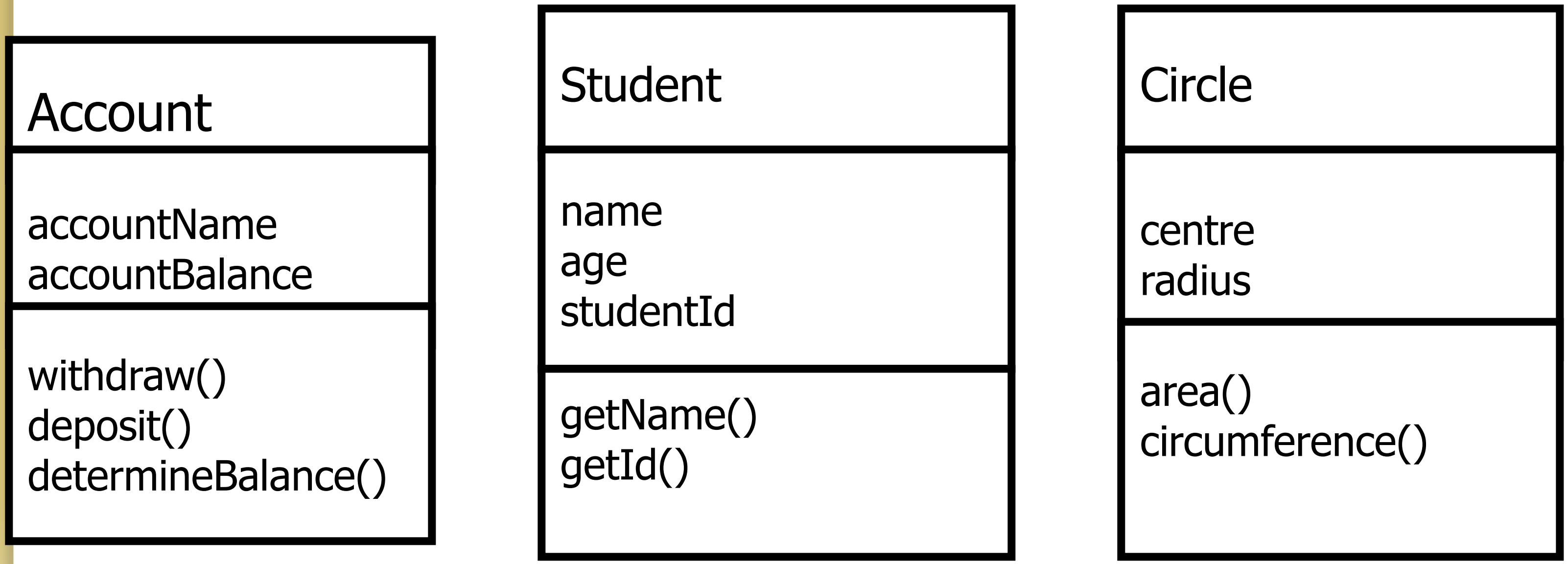
Abstract Into →

Polygon Class
Attributes: `Vertices, Border,`
`Color, FillColor`
Operations: `Draw(), Erase(), Move()`

Figure 1.12: Objects and classes

Class

- Class is a set of *attributes* and *operations* that are performed on the attributes.



Account

accountName
accountBalance

withdraw()
deposit()
determineBalance()

Student

name
age
studentId

getName()
getId()

Circle

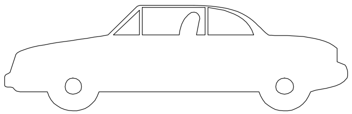
centre
radius

area()
circumference()

Objects

- An Object Oriented system is a collection of interacting Objects.
- Object is an instance of a class.

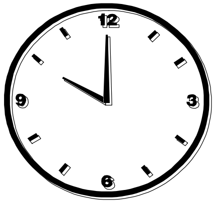
Examples of Objects



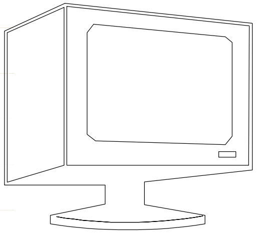
CAR

BOY

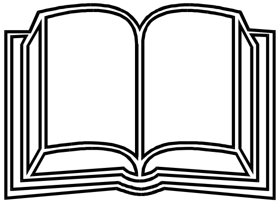
GIRL



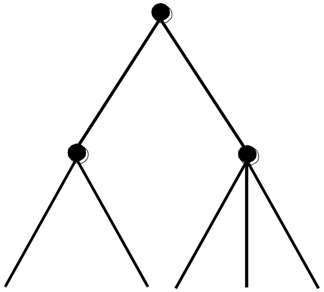
CLOCK



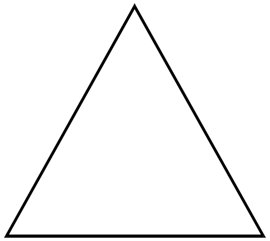
VDU



BOOK



TREE

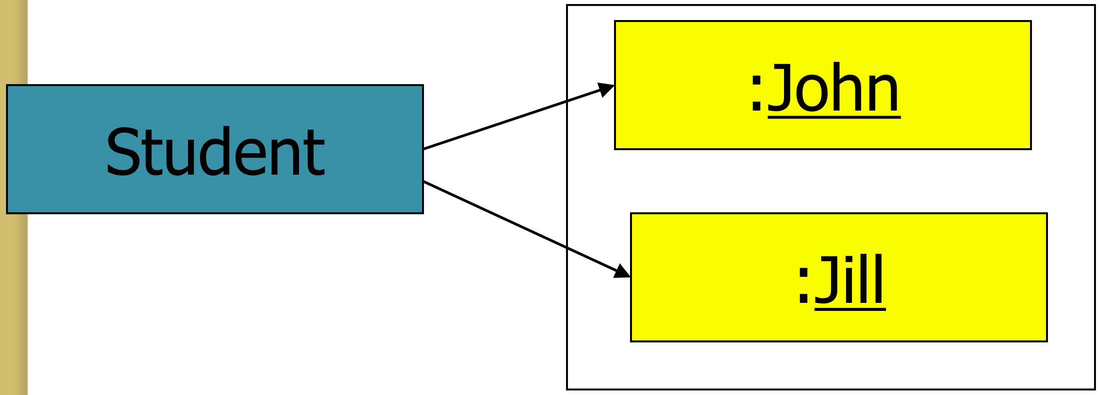


TRIANGLE

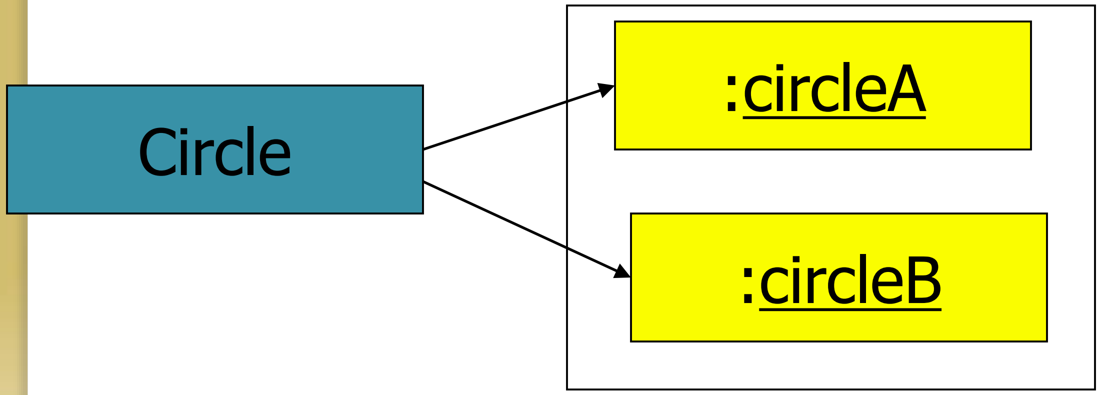
Classification of objects

- ***User Interface objects***
  - Objects that the user interacts directly with
- ***Operating environment objects***
  - Provide services to other components
- ***Task Related objects***
  - Documents, multimedia, problem domain

Classes/Objects



John and Jill are objects of class Student



circleA and circleB are
objects of class Circle

Object

- Objects have state and classes don’t.

  John is an object (instance) of class Student.

  name = “John”, age = 20, studentId = 1236

  Jill is an object (instance) of class Student.

  name = “Jill”, age = 22, studentId = 2345

  circleA is an object (instance) of class Circle.

  centre = (20,10), radius = 25

  circleB is an object (instance) of class Circle.

  centre = (0,0), radius = 10

Basic OOP terminology

Object. contains **data** and **instructions**

Class. **blueprint** for an **object**

Attribute. describe the **state** of objects

Data Type. describes what **kind** of **information** a certain attribute is

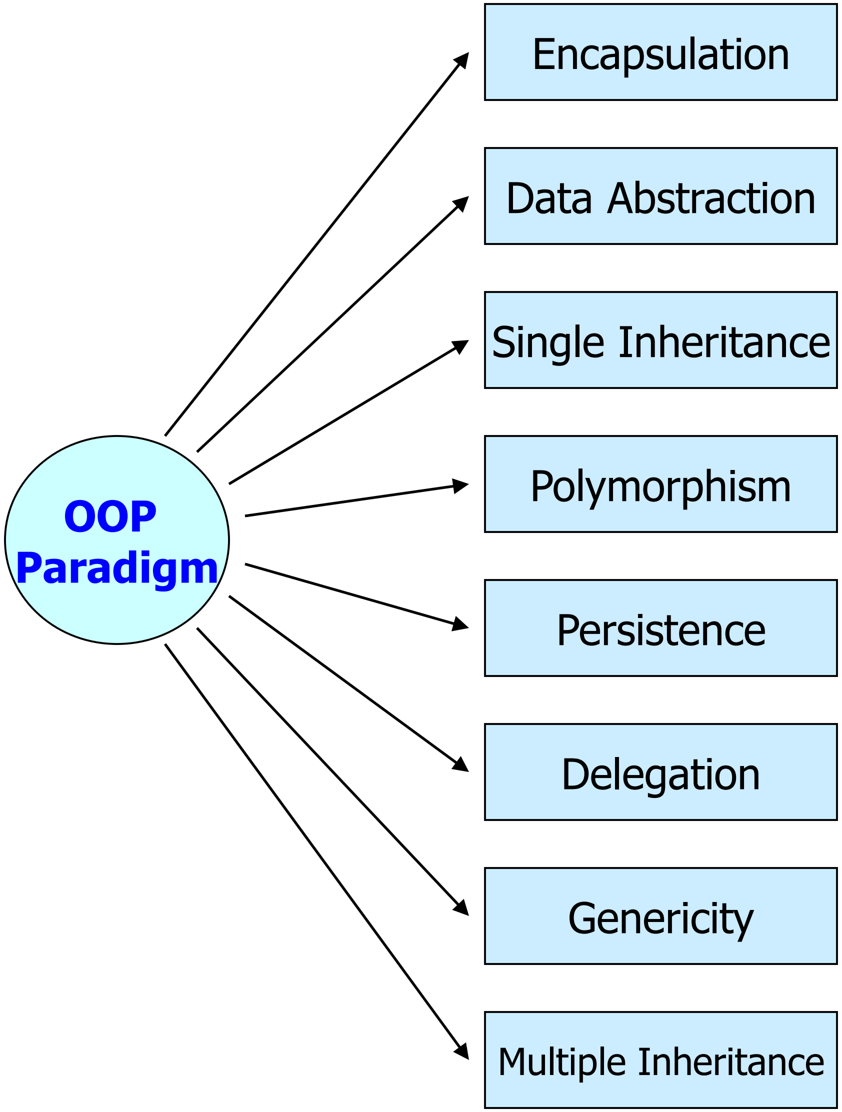
Behavior. **describe** what objects can **do**

Method. a **set** of **instructions**

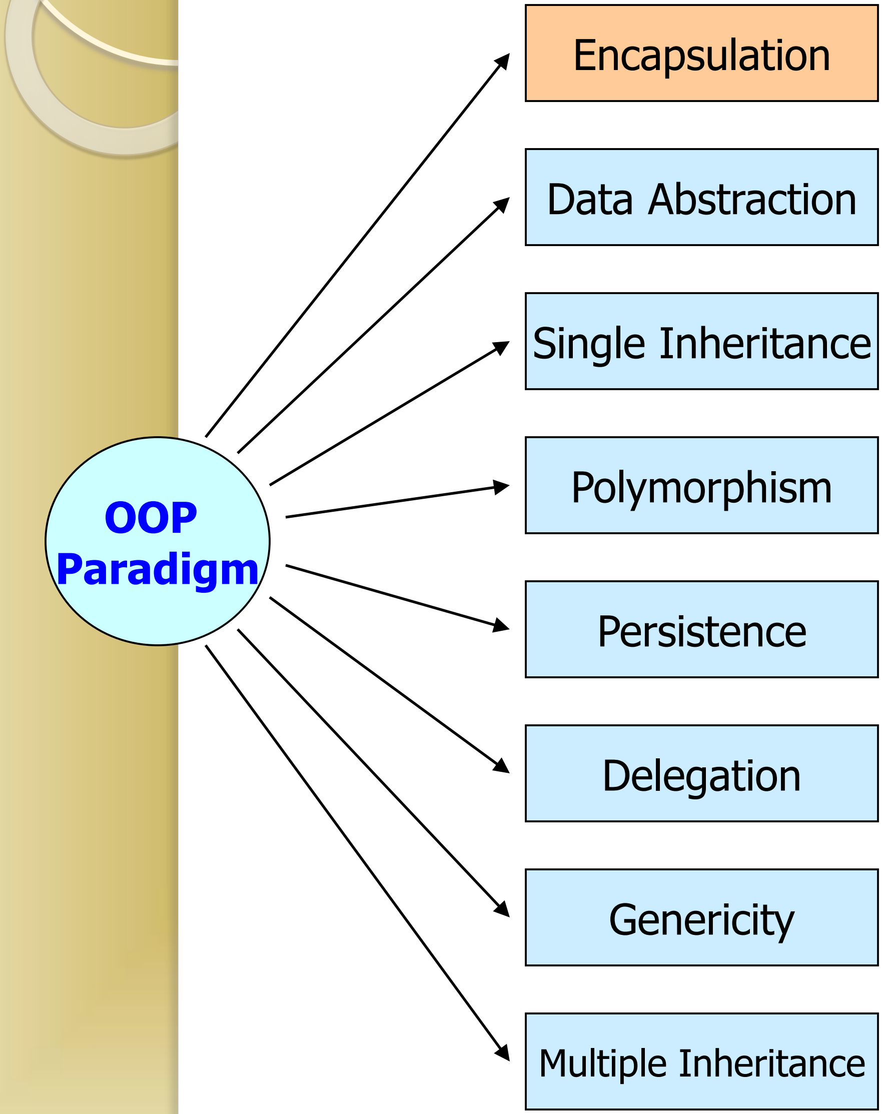
Inheritance. Some objects derive **attributes** and **behaviors** from **other** objects

Encapsulation. **Combining** data and methods together

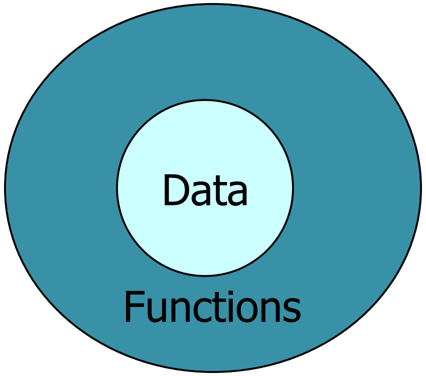
Object Oriented Paradigm: Features



Encapsulation



- It associates the code and the data it manipulates into a single unit; and keeps them safe from external interference and misuse.

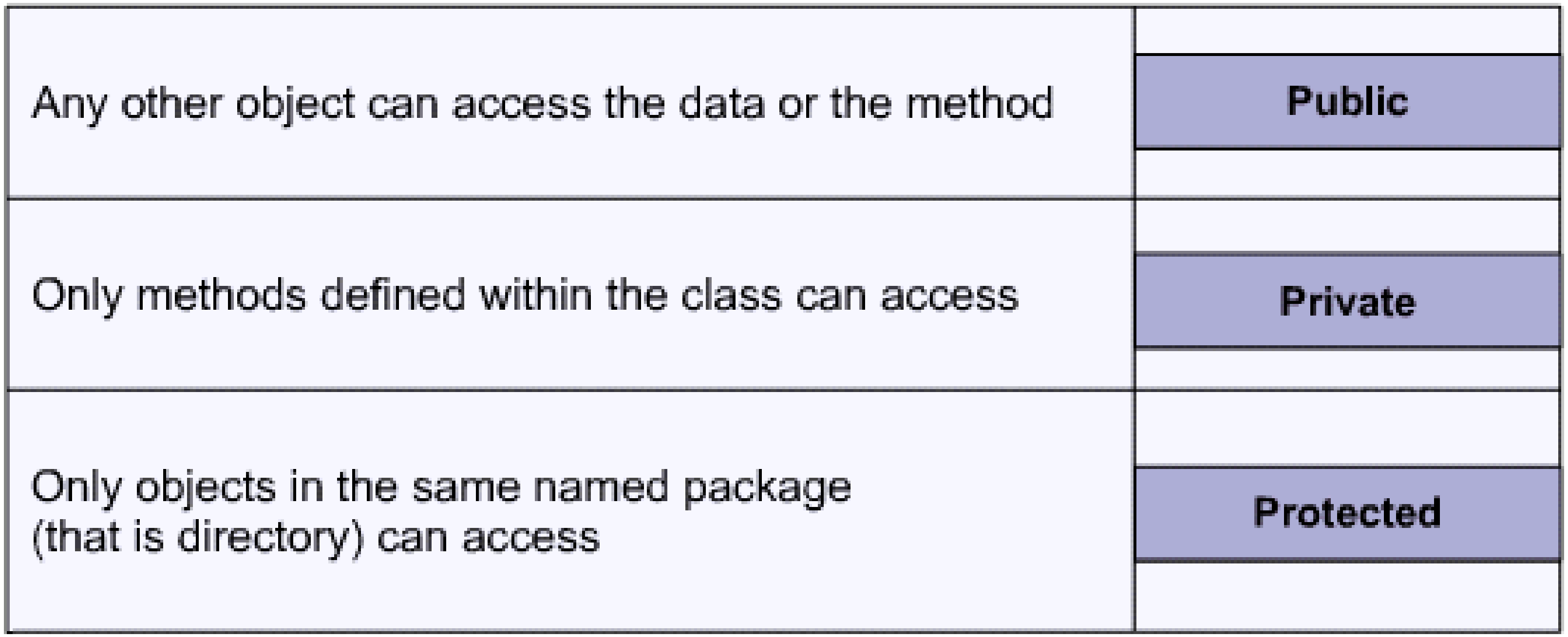


Encapsulation

- All information (attributes and methods) in an object oriented system are stored within the object/class.
- Information can be manipulated through operations performed on the object/class – interface to the class. Implementation is hidden from the user.
- Object support *Information Hiding* – Some attributes and methods can be hidden from the user.

Encapsulation

- To **hide** the details, package **together**
- Access modifiers – **public**, **private** and **protected**

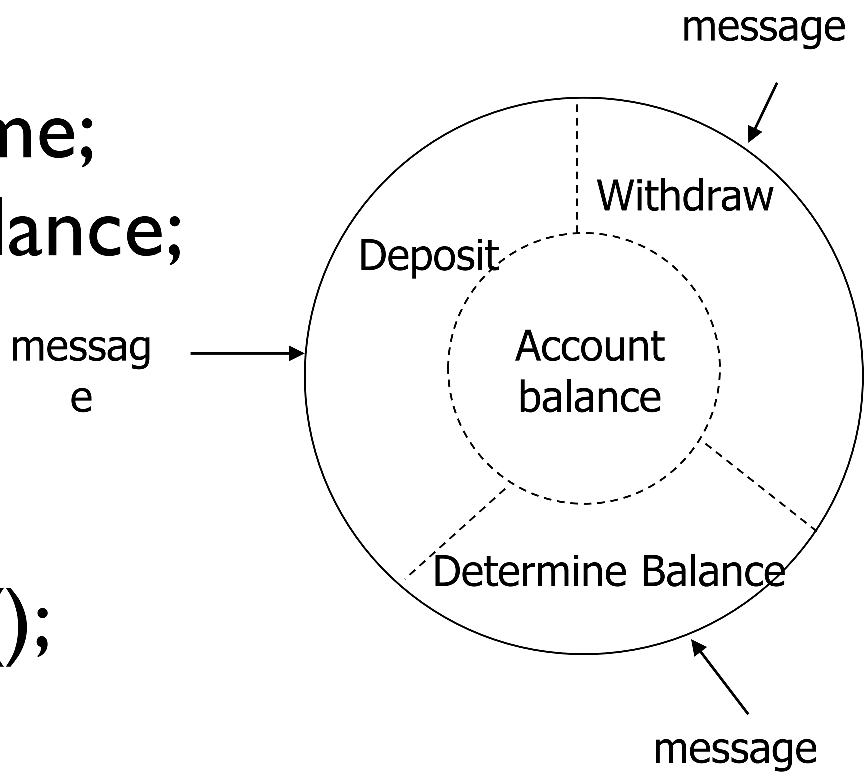


Any other object can access the data or the method	Public
Only methods defined within the class can access	Private
Only objects in the same named package (that is directory) can access	Protected

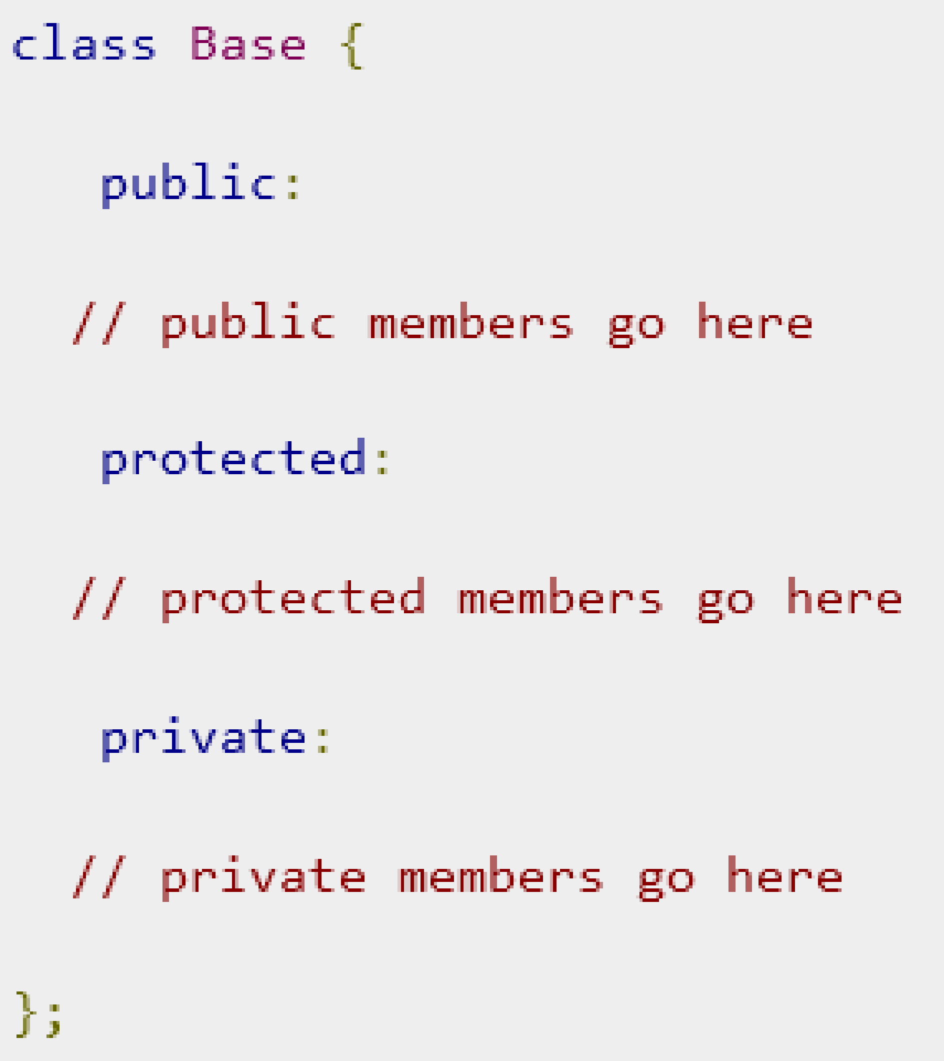
Капсулация на данни - пример

```
class Account {
  private String accountName;
  private double accountBalance;

   public  withdraw();
  public  deposit();
  public  determineBalance();
} // Class Account
```

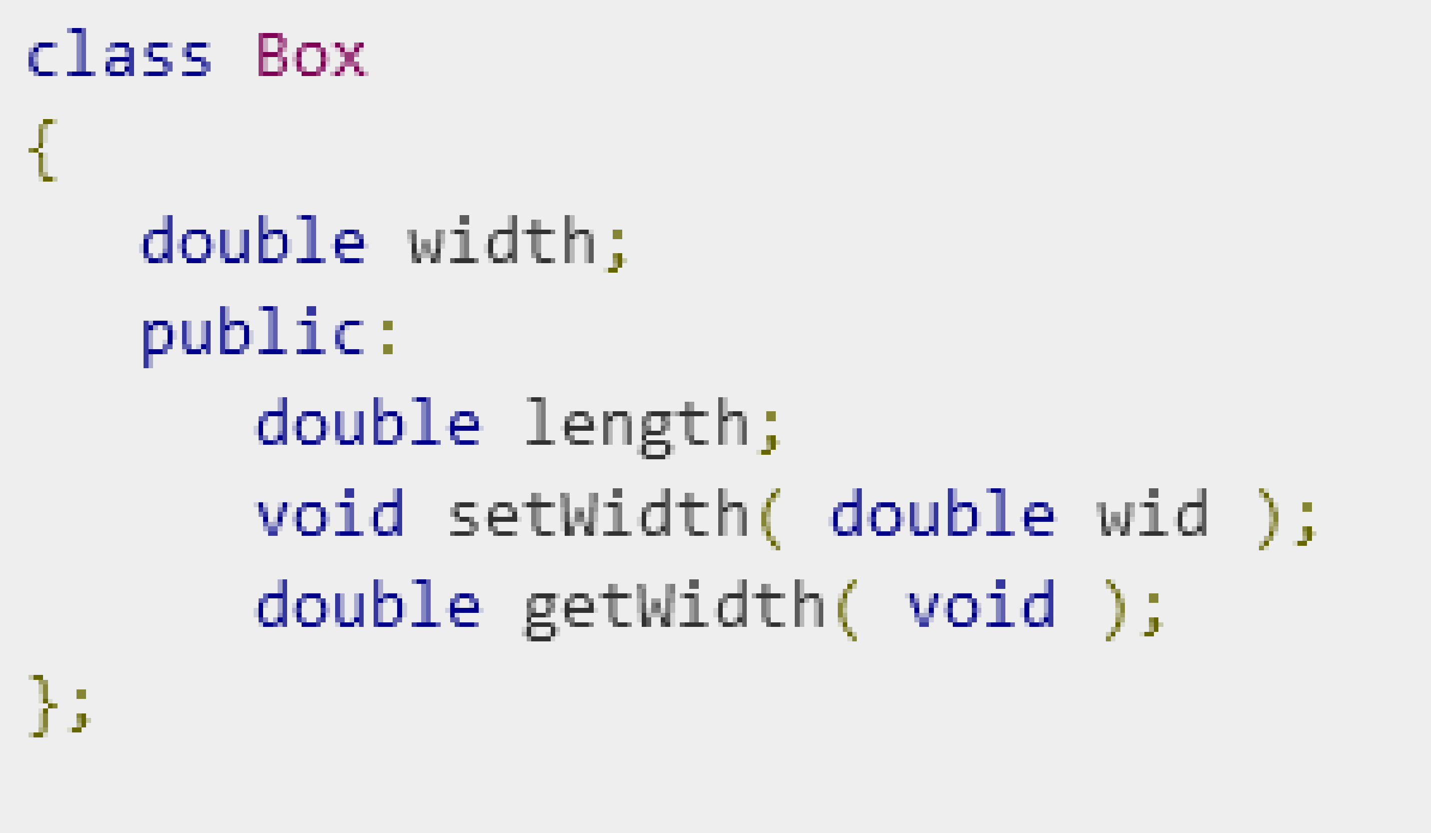


Дефиниция на клас (C++)



```
class Base {
   public:
  // public members go here
   protected:
  // protected members go here
   private:
  // private members go here
};
```

Пример



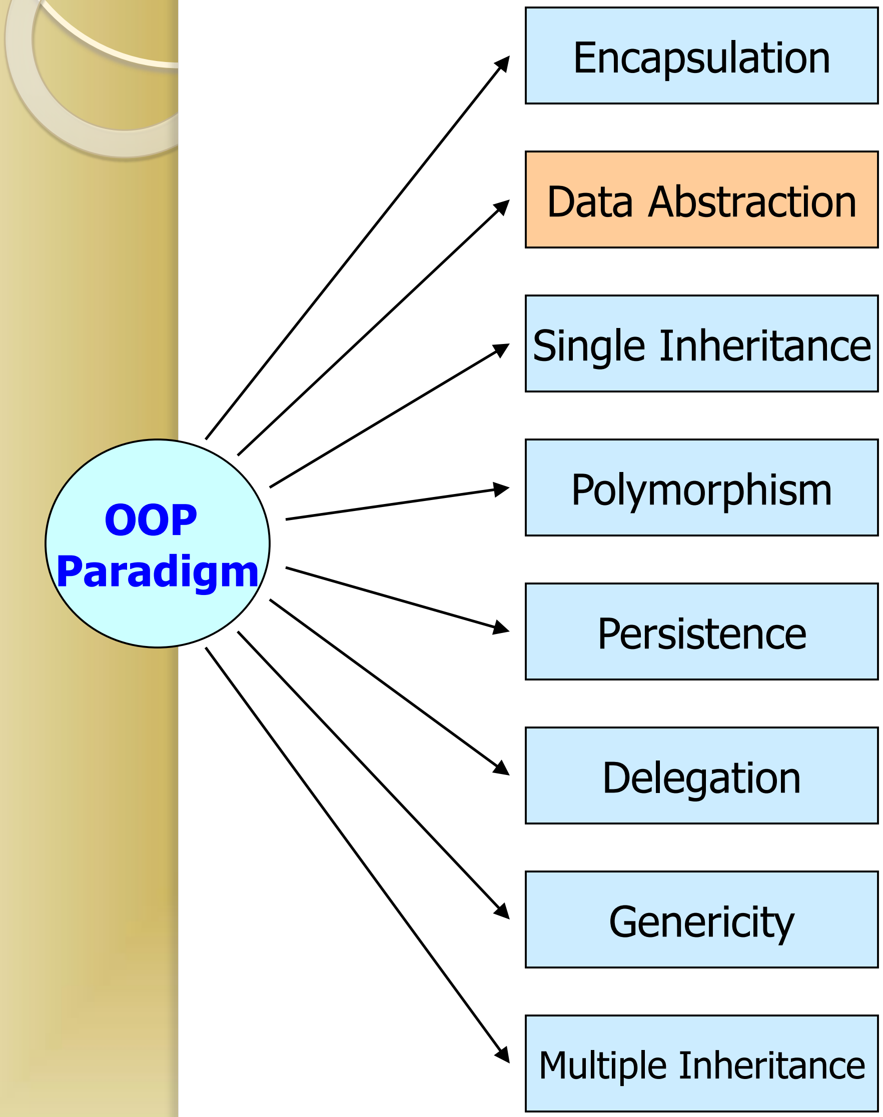
```
class Box
{
   double width;
   public:
      double length;
      void setWidth( double wid );
      double getWidth( void );
};
```

Class- Example

```
class Account {
   private String accountName;
   private double accountBalance;

    public  withdraw();
   public  deposit();
   public  determineBalance();
} // Class Account
```

Data Abstraction



- The technique of creating new data types that are well suited to an application.
- It allows the creation of user defined data types, having the properties of built data types and a set of permitted operators.
- In Java, partial support.
- In C++, fully supported (e.g., operator overloading).

Abstract Data Type (ADT)

- A structure that contains both data and the actions to be performed on that data.

- *Class* is an implementation of an Abstract Data Type.

Data Abstraction

- The technique of creating new data types that are well suited to an application.
- It allows the creation of user defined data types, having the properties of built in data types and more.

Abstraction - Example

cla

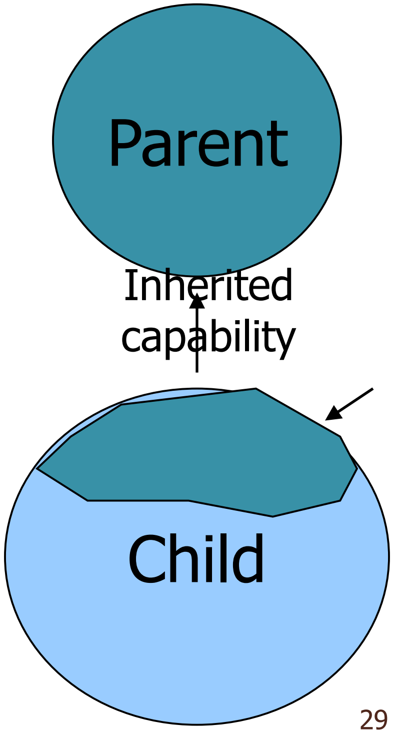
} //

Creates a data type Account

Account acctX;

Inheritance

- New data types (classes) can be defined as extensions to previously defined types.
- Parent Class (Super Class) – Child Class (Sub Class)
- Subclass inherits properties from the parent class.

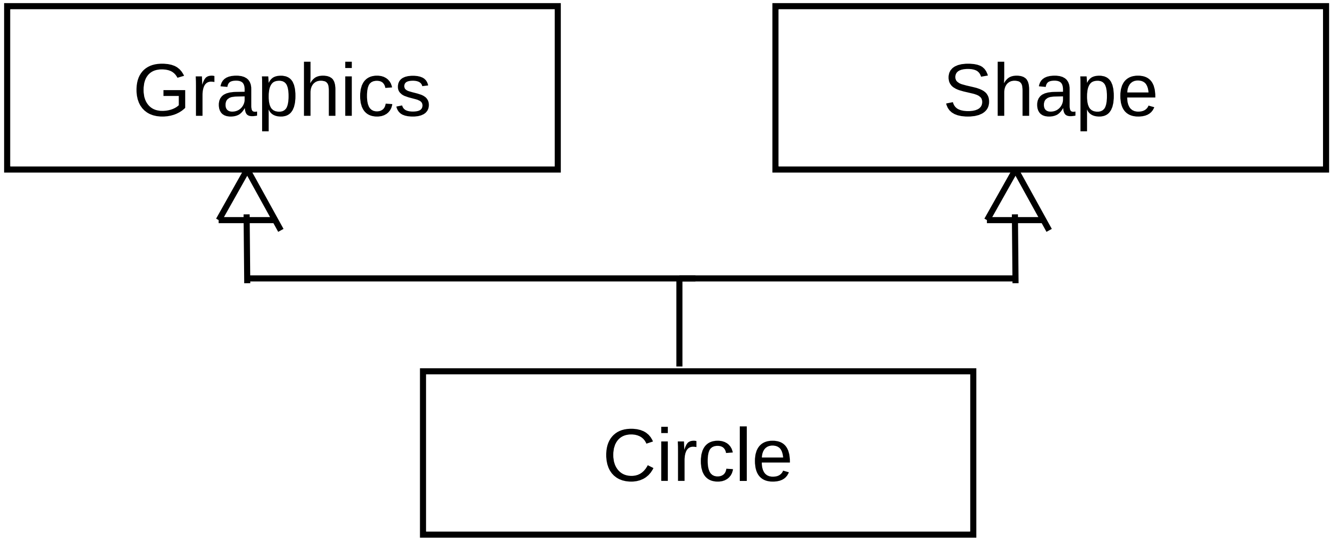


Inheritance - Example

- Example
  - Define Person to be a *class*
    - A Person has *attributes*, such as age, height, gender
    - Assign values to attributes when describing object
  - Define student to be a *subclass* of Person
    - A student has all attributes of Person, plus attributes of his/her own (student no, course_enrolled)
    - A student has all attributes of Person, plus attributes of his/her own (student no, course_enrolled)
    - A student *inherits* all attributes of Person
  - Define lecturer to be a *subclass* of Person
    - Lecturer has all attributes of Person, plus attributes of his/her own (staff_id, subjectID1, subjectID2)

Uses of Inheritance – Multiple Inheritance

- Inherit properties from more than one class.
- This is called *Multiple Inheritance.*



Polymorphism

- Polymorphic which means "many forms" has Greek roots.
  - Poly – many
  - Morphos - forms.
- In OO paradigm polymorphism has many forms.
- Allow a single *object, method, operator* associated with different meaning depending on the type of data passed to it.

Persistence

- The phenomenon where the object outlives the program execution.
- Databases support this feature.

Why OOP?

- *Greater Reliability*
  - Break complex software projects into small, self-contained, and modular objects
- *Maintainability*
  - Modular objects make locating bugs easier, with less impact on the overall project
- *Greater Productivity through Reuse!*
- *Faster Design and Modelling*

Benefits of OOP..

- Inheritance: Elimination of Redundant Code and extend the use of existing classes.
- Build programs from existing working modules, rather than having to start from scratch. → save development time and get higher productivity.
- Encapsulation: Helps in building secure programs.

Benefits of OOP..

- Multiple objects to coexist without any interference.
- Easy to map objects in problem domain to those objects in the program.
- It is easy to partition the work in a project based on objects.
- The Data-Centered Design enables us in capturing more details of model in an implementable form.

Benefits of OOP..

- Object Oriented Systems can be easily upgraded from small to large systems.
- Message-Passing technique for communication between objects make the interface descriptions with external systems much simpler.
- Software complexity can be easily managed.

References

- Robert Grimm, G22.2110-001 Programming Languages, NYU's Computer Science Department
- Prof. Saman Amarasinghe", Prof. Martin Rinard, MIT Course 6.035 Computer Language Engineering
- "Programming with Java" by Balagurusamny, TMH, New Delhi, India.
- "Mastering C++" by V. Rajuk and R. Buyya, Tata McGraw Hill, New Delhi, India.