

Object-oriented paradigm

Object-oriented paradigm

In general, programs are instructions to do something on given data. Programs' two main components are:

- *Data* (or variables to store data).
- Statements that define *computation* (or something to do).

Object-oriented paradigm

- In conventional programming, data and computation are defined separately.
- In object-oriented programming, data and computation must be defined together.

What is an object?

An object is essentially data equipped with relevant computation. Formally, an *object* is defined by:

- A set of attributes that define data.
- A set of operations (or behaviors) that define actions on data.

What is an object?

Example. Lucy, a dog, is an object defined by the following:

- Attributes: female, Golden Retriever, 22 in., 60 lbs., 5 years old.
- Operations (or behaviors): bark, walk, run, sit, jump, swim, ...

What is an object?

Example. Sasha, a dog, is an object defined by the following:

- Attributes: male, German Shephard, 25 in., 70 lbs., 3 years old.
- Operations (or behaviors): bark, walk, run, sit, jump, swim, ...

Objects

Many objects share common attribute types and operations. For example, all dogs share:

- Attributes: gender, breed, height, weight, age.
- Operations (or behaviors): bark, walk, run, sit, jump, swim, ...

In fact, all dog objects can be defined by these attributes and operations.

What is a class?

A class is essentially an abstract definition of objects that share common attribute types and operations. Formally, a *class* is defined by:

- A set of attributes without specific data.
- A set of operations (or behaviors) that define actions on attributes.

Classes and objects

- A class is an abstract description of objects that share common attribute types and operations.
- A class may be regarded as simply collection of objects that share common attribute types and operations.
- An object is an instance of a class. For example, the object Lucy is an instance of the class Dog.

Classes and objects

- In Java, an object must be defined as an instance of a class.
- Every Java class has a *constructor* that instantiates an object.
- In Java, attributes are called *fields*, and operations (or behaviors) are called *methods*.

Object-oriented program development cycle

Object-oriented program development cycle

It is a five-step cycle consisting of:

- Design the program
- Write the code
- Correct syntax errors
- Test the program
- Correct semantic (or logic) errors

Design the program

Analyze and formalize the problem to be solved. Determine a class to model and write a *class specification*:

- a list of attributes
- a list of behaviors

This gives an overall outline of a class.

Design the program

A *UML (Unified Modeling Language) diagram* is often used for a class specification:

Riddle
<ul style="list-style-type: none">– question : String– answer : String
<ul style="list-style-type: none">+ Riddle(q : String, a : String)+ getQuestion() : String+ getAnswer() : String

Design the program

- For each behavior, specify parameters and what to return (if any), and describe an algorithm in pseudocode.
- This gives outlines of methods.

Write the code

- Translate your design into Java code.
- More precise your design is, easier it is to translate: One line of pseudocode should correspond to one line of program code.

Java class

Java class

```
public class Dog {
```

Fields (attributes)

Constructor

Methods (operations or behaviors)

```
}
```

Accessibility

Classes, fields and methods must be declared with *access specifiers* (or *access modifiers*).

- Classes and methods are usually declared as **public** — accessible by anyone from anywhere.
- Fields are usually declared as **private** — accessible only within the object.

Defining fields

```
public class Dog {  
  
    private String breed;  
    private char gender;  
    private double height;  
    private double weight;  
    private int age;
```

Constructor

Methods (operations or behaviors)

```
}
```

Constructor

- A *constructor* is a special method that instantiates an object.
- Objects are created from a class by invoking a constructor.
- Constructors usually have parameters, and constructors create objects according to the values of parameters.

Defining a constructor

```
public class Dog {  
  
    private String breed;  
    private char gender;  
    private double height;  
    private double weight;  
    private int age;  
  
    public Dog(String b, char g, double h, double w, int a) {  
        breed = b;  
        gender = g;  
        height = h;  
        weight = w;  
        age = a;  
    }  
  
    Methods (operations or behaviors)  
  
}
```

Methods

- *Methods* define objects' operations, similar to the way functions define basic computational tasks.
- Many (but not all) methods return values, just like many functions return values.
- All methods be declared with return types (which specify the types of values being returned).

Defining methods

```
public class Dog {  
  
    private String breed;  
    private char gender;  
    private double height;  
    private double weight;  
    private int age;
```

Constructor

```
    public void getOlder() {  
        age++;  
    }
```

```
}
```

Defining methods

```
public class Dog {  
  
    private String breed;  
    private char gender;  
    private double height;  
    private double weight;  
    private int age;
```

Constructor

```
    public int getAge() {  
        return age;  
    }
```

```
}
```

Instantiating an object

```
Dog lucy = new Dog("Golden Retriever",  
    'f', 22.0, 60.0, 5);
```

```
lucy.getOlder();  
System.out.print("Lucy is now " +  
    lucy.getAge() + " years old.");
```