

Java classes

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

- *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;  
    }
```

```
}
```

Calling methods

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

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

Accessors

- A method that simply returns the value of a field is called an *accessor* (or "*get*" *method*).
- An accessor for a field f is usually named `getF`.
- An accessor has no parameter.
- The return type of an accessor for a field f is the type of f .

Accessors

```
public class Coo {  
    private type f;  
  
    ...  
  
    public type getF() {  
        return f;  
    }  
}
```

Mutators

- A method that simply changes the value of a field by a parameter value is called a *mutator* (or *modifier* or "*set*" *method*).
- A mutator for a field f is usually named `setF`.
- A mutator returns nothing.
- A mutator for a field f has a parameter whose type is the same as that of f .

Mutators

```
public class Coo {  
  
    private type f;  
  
    ...  
  
    public type getF() {  
        return f;  
    }  
  
    public void setF(type f0) {  
        f = f0;  
    }  
  
}
```

Method calls

Every method belongs to some object. In general, when we call a method, we must specify both the object and method:

```
object.method(<arguments>)
```

Internal method calls

To invoke a method within the same object, we can just call by the method name.

```
public class Dog {  
  
    public void bark(String b) {  
        ...  
    }  
  
    public void run() {  
        bark("Woof!");  
        ...  
    }  
}
```

Internal method calls

Or, to be more explicit, we may also put "this" with the method name.

```
public class Dog {  
  
    public void bark(String b) {  
        ...  
    }  
  
    public void run() {  
        this.bark("Woof!");  
        ...  
    }  
}
```

External method calls

To invoke a method of another object, we must always explicitly specify the object.

In general, from a class `Doo`, to call a `Coo` object `c`'s method `noo()`, we put:

```
public class Doo {  
  
    public void poo() {  
        Coo c = new Coo();  
        c.noo(7);  
    }  
  
}
```