Main Principles of OOP in Java and C++

Diana Akolzina

May 16, 2023

1 Introduction

Object-Oriented Programming (OOP) is a programming paradigm that uses "objects" to design applications and software. There are four main principles of OOP: encapsulation, inheritance, polymorphism, and abstraction.

2 Encapsulation

Encapsulation is defined as the wrapping up of data under a single unit. It is the mechanism that binds together code and the data it manipulates.

2.1 Java

```java
public class Encapsulation {
    private String name;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

2.2 C++

```cpp
class Encapsulation {
    private:
        std::string name;

    public:
        void setName(std::string name) {
```

```
        this->name = name;
    }

    std::string getName() {
        return name;
    }
};
```

3 Inheritance

Inheritance is a process of object reusability. It is a mechanism where a new class is derived from an existing class.

3.1 Java

```
public class Base {
}

public class Derived extends Base {
}
```

3.2 C++

```
class Base {
};

class Derived: public Base {
};
```

4 Polymorphism

Polymorphism allows us to perform a single action in different ways. It provides an ability to a class to have multiple implementations with the same name.

4.1 Java

```
public class Animal {
    public void sound() {
    }
}

public class Dog extends Animal {
```

```
    public void sound() {
        System.out.println("Bark");
    }
}
```

4.2 C++

```
class Animal {
    public:
        virtual void sound() {
        }
};

class Dog: public Animal {
    public:
        void sound() {
            std::cout << "Bark";
        }
};
```

5 Abstraction

Abstraction is a process where you show only "relevant" data and "hide" unnecessary details of an object from the user.

5.1 Java

```
public abstract class Animal {
    public abstract void sound();
}
```

5.2 C++

```
class Animal {
    public:
        virtual void sound() = 0;  // Pure virtual function
};
```