HomeWork1

1. 什么是 IoC（Inversion of Control）、DIP（Dependency Inversion Principle）、DI（Dependency Injection）？

这三者的关系是：**DIP 是一种软件架构设计原则，IoC 是 DIP 的一种实现方式，DI 是 IoC 的实现方式**

（1）DIP：**依赖倒置原则**

依赖倒置原则，它转换了依赖，高层模块不依赖于低层模块的实现，而低层模块依赖于高层模块定义的接口。通俗的讲，就是高层模块定义接口，低层模块负责实现。这样，当有新的低层模块实现时，不需要修改高层模块的代码。因此可以达到以下三个目的：

1. 系统更柔韧：可以修改一部分代码而不影响其他模块。
2. 系统更健壮：可以修改一部分代码而不会让系统崩溃。
3. 系统更高效：组件松耦合，且可复用，提高开发效率。

（2）IoC：**控制反转原则**

DIP 是一种软件设计原则，仅展现了两个模块之间应该如何依赖，但没有说怎么去实现

IoC 是一种软件设计模式，它告诉你应该如何做，来解除相互依赖模块的耦合。控制反转（IoC），它为相互依赖的组件提供抽象，**将依赖（低层模块）对象的获得交给第三方（系统）来控制**，即依赖对象不在被依赖模块的类中直接通过 new 来获取。

IoC 有两种常见的实现方式：依赖注入、服务定位。依赖注入（DI）使用最为广泛

（3）DI：**依赖注入**

控制反转 IoC 是一种重要的软件设计模式，就是将依赖对象的创建和绑定转移到被依赖对象类的外部来实现。依赖注入 DI 则提供一种机制，将需要依赖（低层模块）对象的引用传递给被依赖（高层模块）对象。

依赖注入有三种主要方式：

① 构造函数注入

构造函数函数注入，毫无疑问通过构造函数传递依赖。因此，构造函数的参数必然用来接收一个依赖对象。根据 DIP 原则，我们知道高层模块不应该依赖于低层模块，两者应该依赖于抽象。那么构造函数的参数应该是一个抽象类型。

② 属性注入

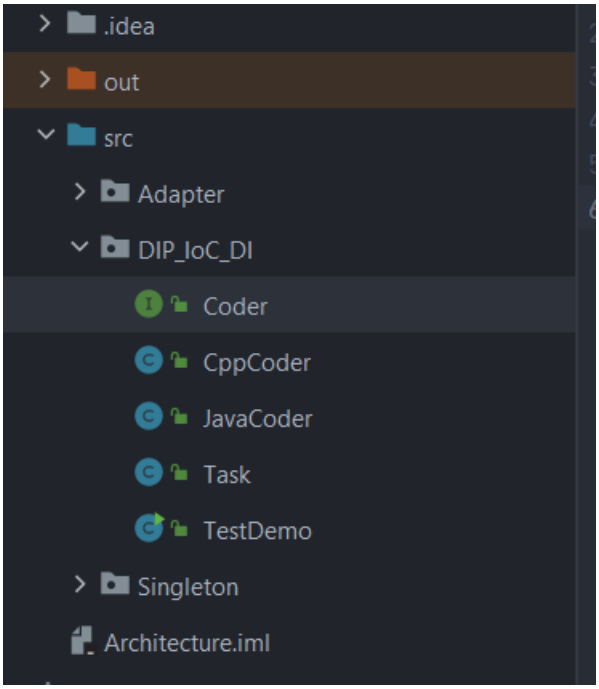
即通过属性来传递依赖

③ 接口注入

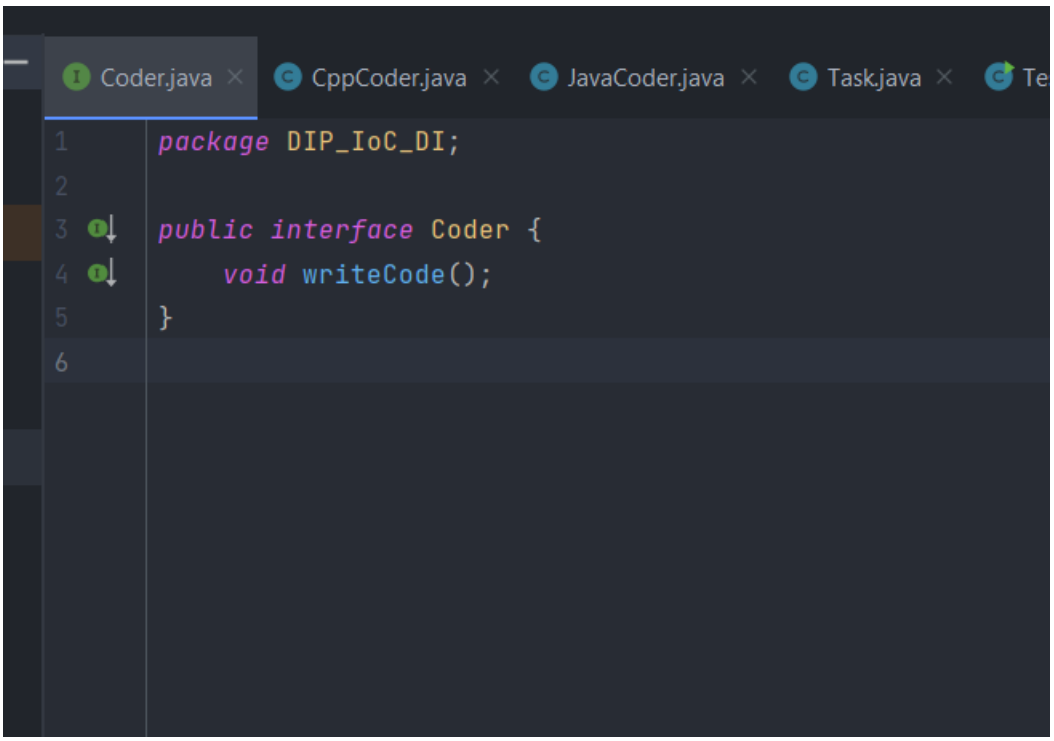
相比构造函数注入和属性注入，接口注入显得有些复杂，使用也不常见。具体思路是先定义一个接口，包含一个设置依赖的方法。然后依赖类，继承并实现这个接口。

2. 举例说明实现方式

代码结构如下（DIP_IoC_DI）：



高层模块定义 Coder 的 writeCode 接口



低层模块 JavaCoder、CppCoder 负责实现

```java
package DIP_IoC_DI;

public class JavaCoder implements Coder {
    private String name;
    public JavaCoder(String name) {
        this.name = name;
    }

    @Override
    public void writeCode() {
        System.out.println("Now Coder " + this.name + " is writing code in Java");
    }
}
```

Coder.java × CppCoder.java × JavaCoder.java × Task.java × TestDemo.java ×

```java
package DIP_IoC_DI;

public class CppCoder implements Coder {
    private String name;
    public CppCoder(String name) {
        this.name = name;
    }

    @Override
    public void writeCode() {
        System.out.println("Now Coder " + this.name + " is writing code in C++");
    }
}
```

编写 Task 类来具体使用 Coder 模块

```java
package DIP_IoC_DI;

public class Task {
    private String taskName;
    private Coder coder;
    public Task(String taskName) {
        this.taskName = taskName;
    }

    public void setCoder(Coder coder) {
        this.coder = coder;
    }

    public void startTask() {
        System.out.println("Now Task [" + this.taskName + "] has started");
        this.coder.writeCode();
    }
}
```

测试类及运行结果:

```java
package DIP_IoC_DI;

public class TestDemo {
    public static void main(String [] args) {
        Task task1 = new Task( taskName: "MiHoYoDestruction");
        task1.setCoder(new JavaCoder( name: "Zhong Li"));
        task1.startTask();

        Task task2 = new Task( taskName: "MiHoYoConstruction");
        task2.setCoder(new CppCoder( name: "Ke Qing"));
        task2.startTask();
    }
}
```

```
TestDemo
"C:\Program Files\Java\jdk-11.0.11\bin\java.exe" "-javaagent:E:\IntelliJ IDEA 2021.
 E:\java\Architecture\out\production\Architecture DIP_IoC_DI.TestDemo
Now Task [MiHoYoDestruction] has started
Now Coder Zhong Li is writing code in Java
Now Task [MiHoYoConstruction] has started
Now Coder Ke Qing is writing code in C++

Process finished with exit code 0
```

Homework2

1. 针对 Iterator 的例子，将存储 Book 用的数组换成其他 Collection 并运行

将数组替换为 ArrayList，其余同课件例

```java
import java.util.ArrayList;

public class BookShelf implements Aggregate {
    private ArrayList<Book> books;
    private int length;
    public BookShelf(int maxsize) {
        this.books = new ArrayList<>(maxsize);
        this.length = 0;
    }

    public Book getBookAt(int index) {
        return books.get(index);
    }

    public int getLength() {
        return this.length;
    }

    public void appendBook(Book book) {
        this.books.add(book);
        length++;
    }

    public Iterator iterator() {
        return new BookShelfIterator(this);
    }
}
```

测试代码:

```java

public class testDemo {
    public static void main(String [] args) {
        BookShelf bookShelf = new BookShelf( maxsize: 5);
        bookShelf.appendBook(new Book( name: "C primer plus"));
        bookShelf.appendBook(new Book( name: "Java How to Program"));
        bookShelf.appendBook(new Book( name: "Thinking in Java"));
        bookShelf.appendBook(new Book( name: "Journey to the West"));
        bookShelf.appendBook(new Book( name: "Three Kingdoms"));
        bookShelf.appendBook(new Book( name: "深入理解计算机系统"));

        Iterator it = bookShelf.iterator();
        while (it.hasNext()) {
            Book book = (Book) it.next();
            System.out.println(book.getName());
        }
    }
}

```

运行结果:

```
"C:\Program Files\Java\jdk-11.0.11\bin\java.exe" "-javaagent:E:\IntelliJ IDEA 2021.1
 E:\java\Architecture\out\production\Architecture Iterator.BookShelf.testDemo
C primer plus
Java How to Program
Thinking in Java
Journey to the West
Three Kingdoms
深入理解计算机系统

Process finished with exit code 0
```

2. 设计一个 Specified 的 Iterator 并运行

设计一个用来遍历点名表的迭代器，点名表中是学生。迭代器用来根据学生的出勤情况来进行确认，然后打印出学生的签到情况

```java
package Iterator.Specified;

public class Student {
    private String name;
    private String sNo;
    private String grade;
    private String major;
    private Boolean isPresent;
    private Boolean isChecked;

    public Student(String name, String sNo, String grade, String major) {
        this.name = name;
        this.sNo = sNo;
        this.grade = grade;
        this.major = major;
        this.isPresent = false;
        this.isChecked = false;
    }

    public String getName() {
        return name;
    }

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

```java
public class CheckList implements Aggregate{

    private ArrayList<Student> studentCheckList;
    private int length;

    public CheckList(int maxsize) {
        this.studentCheckList = new ArrayList<>(maxsize);
        this.length = 0;
    }

    public void appendStudent(Student student) {
        this.studentCheckList.add(student);
        length++;
    }

    public int getTotal() {
        return length;
    }

    public Student getStudentAtIndex(int index) {
        return studentCheckList.get(index);
    }

    public Iterator iterator() {
        return new CheckListIterator(this);
    }
}
```

```java
public class CheckListIterator implements Iterator {
    private CheckList checkList;
    private int index;

    public CheckListIterator(CheckList checkList) {
        this.checkList = checkList;
        this.index = 0;
    }

    @Override
    public boolean hasNext() {
        return index < checkList.getTotal();
    }

    @Override
    public Object next() {
        Student student = checkList.getStudentAtIndex(index);
        index++;
        return student;
    }
}
```

测试代码如下:

设置学生 3 4 未出席，其余学生出席

```java
package Iterator.Specified;

public class testDemo {
    public static void main(String [] args) {
        CheckList checkList = new CheckList( maxsize: 5);
        Student student1 = new Student( name: "yjz",  sNo: "1",  grade: "2019",  major: "se");
        student1.setPresent(true);
        Student student2 = new Student( name: "xjx",  sNo: "2",  grade: "2018",  major: "se");
        student2.setPresent(true);
        Student student3 = new Student( name: "xsr",  sNo: "3",  grade: "2020",  major: "cs");
        Student student4 = new Student( name: "cht",  sNo: "4",  grade: "2021",  major: "se");
        Student student5 = new Student( name: "lhz",  sNo: "5",  grade: "2019",  major: "se");
        student5.setPresent(true);

        checkList.appendStudent(student1);
        checkList.appendStudent(student2);
        checkList.appendStudent(student3);
        checkList.appendStudent(student4);
        checkList.appendStudent(student5);

        Iterator iterator = checkList.iterator();
        while (iterator.hasNext()) {
            Student student = (Student) iterator.next();
            // 检查学生是否签到
            if (student.getPresent()) {
                student.setChecked(true);
            }
        }

        Iterator iterator1 = checkList.iterator();
        while (iterator1.hasNext()) {
            Student student = (Student) iterator1.next();
            System.out.printf("Name: %-5s  No: %-3s  Grade: %-6s  Major: %-4s  Status: %s%n",
                    student.getName(),
                    student.getsNo(),
                    student.getGrade(),
                    student.getMajor(),
                    student.getChecked()?"Ok":"No");
        }

    }
```

运行结果:
签到状态显示为学生 3 4 未签到

```
"C:\Program Files\Java\jdk-11.0.11\bin\java.exe" "-javaagent:E:\IntelliJ IDEA 202
 E:\java\Architecture\out\production\Architecture Iterator.Specified.testDemo
Name: yjz    No: 1    Grade: 2019    Major: se    Status: Ok
Name: xjx    No: 2    Grade: 2018    Major: se    Status: Ok
Name: xsr    No: 3    Grade: 2020    Major: cs    Status: No
Name: cht    No: 4    Grade: 2021    Major: se    Status: No
Name: lhz    No: 5    Grade: 2019    Major: se    Status: Ok

Process finished with exit code 0
```