课 后 作 业[1]

课程名称	Java 应用与开发	开课学期	2018 年秋季学期
班　　级	17 计算机科学与技术二班	姓　　名	石晓晨
学　　号	17020031057	联系方式	18853816845
完成情况	（不要填）		

简答题

1. 分析抽象类和接口的区别,说明抽象类和接口的作用

接口是公开的，里面不能有私有的方法或变量，是用于让别人使用的，而抽象类是可以有私有方法或私有变量的，

另外，实现接口的一定要实现接口里定义的所有方法，而实现抽象类可以有选择地重写需要用到的方法，一般的应用里，最顶级的是接口，然后是抽象类实现接口，最后才到具体类实现。

还有，接口可以实现多重继承，而一个类只能继承一个超类，但可以通过继承多个接口实现多重继承，接口还有标识（里面没有任何方法，如 Remote 接口）和数据共享（里面的变量全是常量）的作用.

2. 什么是泛型？泛型有什么作用

泛型：就是一种不确定的数据类型。

类型的参数化，就是可以把类型像方法的参数那样传递。
泛型使编译器可以在编译期间对类型进行检查以提高类型安全，减少运行时由于对象类型不匹配引发的异常。

省略了强转的代码。
泛型方法，算法的复用

编程题

1. 题目

[1]注意事项（仔细阅读）：电子版提交到系统，无需纸质版（课后开放提交入口，会通过微信通知大家）；作业文件命名格式为“2018-autumn-hw-学号(id)-姓名(name)-完成时间（例如 20180918）.docx”；不能更改作业报告格式和删除格式中的文字，注意实验报告的撰写，注重格式，注意笔误；注意实验报告的命名及撰写也作为考核的一部分。

自行定义一个泛型类，尝试写出代码

Code: Generic.java

```java
package ouc.cs;

public class Generic<T> {
    private T obj;

    public T getObj() {
        return obj;
    }

    public void setObj(T obj) {
        this.obj = obj;
    }
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Generic<String> genericA = new Generic<String>();
        genericA.setObj("Java");
        System.out.println(genericA.getObj());
        Generic<Integer> genericB = new Generic<Integer>();
        genericB.setObj(18);
        System.out.println(genericB.getObj());
    }
}
```

2. 题目

练习泛型化集合类型基于通配符的遍历方法。

Code:VectorTraverse.java

```java
package ouc.cs;

import java.util.Vector;

public class VectorTraverse {
    public void overview(Vector<?> vector){
        for(Object o:vector){
            System.out.println(o);
        }
    }
```

```
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Vector<Integer> vector1 = new Vector<>();
        for(int i=0;i<9;i++){
            vector1.addElement(i);
        }
        Vector<Character> vector2 = new Vector<>();
        for(char i='A';i<'Z';i++){
            vector2.add(i);
        }
        VectorTraverse vectorTraverse = new VectorTraverse();
        vectorTraverse.overview(vector1);
        System.out.println("\n");
        vectorTraverse.overview(vector2);

    }

}
```