原型模式实验

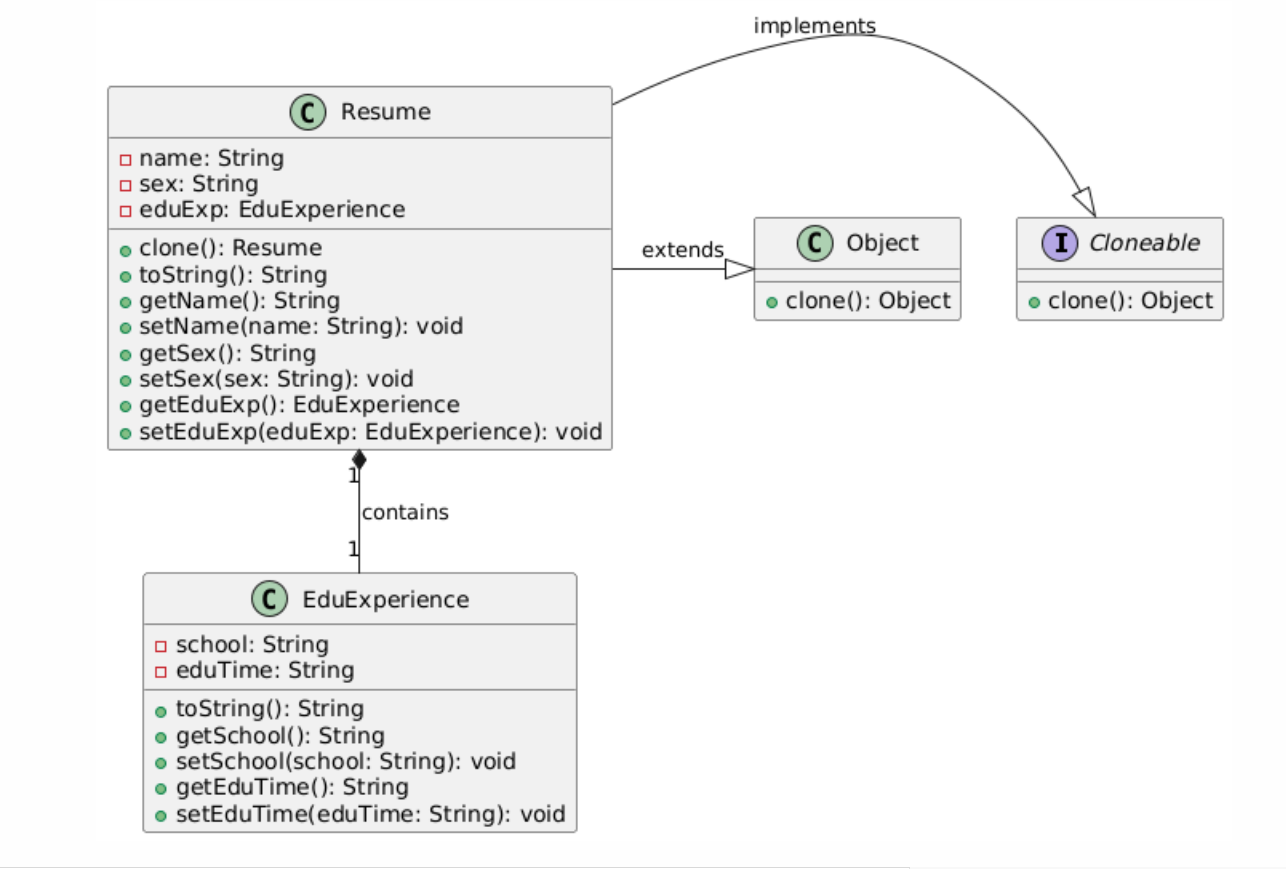
**【实验目的及要求】**

1. 熟悉原型模式的定义；
2. 掌握原型模式角色及作用，熟悉原型模式结构图；
3. 掌握原型模式编程。

**【实验内容】**

**在工作和学习中经常需要填写个人简历。假如个人简历包括基本信息（包括姓名、性别）和学历（学习时间段，学校）。由于每个人有多个学习时段（考虑小学、初中、高中三个阶段）。在程序中，每一个学习时段，都创建一个对象。使用原型模式实现个人简历。**

**【实验步骤】**

1. **个人简历的类图**
2. ****
3. **实现代码**
4. **个人简历类----**Resume

**package** pattern.prototype;

**public** **class** Resume **implements** Cloneable {

**private** String name;

**private** String sex;

**private** EduExperience eduExp;

public Resume(String name, String sex, EduExperience eduExp) {

this.name = name;

this.sex = sex;

this.eduExp = eduExp;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public EduExperience getEduExp() {

return eduExp;

}

public void setEduExp(EduExperience eduExp) {

this.eduExp = eduExp;

}

public String getSex() {

return sex;

}

public void setSex(String sex) {

this.sex = sex;

}

@Override

public Resume clone() {

try {

Resume clone = (Resume) super.clone();

// TODO: copy mutable state here, so the clone can't change the internals of the original

return clone;

} catch (CloneNotSupportedException e) {

throw new AssertionError();

}

}

@Override

public String toString() {

return "基本信息[姓名："+name+" 性别："+sex+"] "

+ "学历[学习时间："+eduExp.getEduTime()+" 学校："+eduExp.getSchool()+"]]";

}}

1. **学习时段类(EduExperience)**

**package** pattern.prototype;

**public** **class** EduExperience {

**private** String school; //学校

**private** String eduTime; //学习时间段

**public EduExperience(String school, String eduTime) {**

**this.school = school;**

**this.eduTime = eduTime;**

**}**

**public String getSchool() {**

**return school;**

**}**

**public void setSchool(String school) {**

**this.school = school;**

**}**

**public String getEduTime() {**

**return eduTime;**

**}**

**public void setEduTime(String eduTime) {**

**this.eduTime = eduTime;**

**}**

**@Override**

**public String toString() {**

**return "EduExperience{" +**

**"school='" + school + '\'' +**

**", eduTime='" + eduTime + '\'' +**

**'}';**

**}**

}

1. **测试类Test**

**package** pattern.prototype;

**public** **class** Test {

**public** **static** **void** main(String[] args) {

//使用克隆方式创建小学、初中、高中三个学历对象

EduExperience litleEduExperience = new EduExperience("希望小学","2010-2016");

EduExperience middleEduExperience = new EduExperience("阳光中学","2016-2019");

EduExperience bigEduExperience = new EduExperience("草莓大学","2019-2023");

oldResume = new Resume("fancy","女",litleEduExperience);

newResume1=oldResume.clone();

newResume1.setEduExp(middleEduExperience);

newResume2=newResume1.clone();

newResume2.setEduExp(bigEduExperience);

//输出显示三个学历的信息

System.out.println("简历1"+oldResume.toString());

System.out.println("简历2"+newResume1.toString());

System.out.println("简历3"+newResume2.toString());

}

**【思考问题】**

1. 你实现的程序是浅克隆还是深克隆？为什么？

浅克隆。因为Object类提供的接口实现类的Cloneale的clone()方法，实现的是浅克隆。

System.*out*.println("简历地址是否相同"+(oldResume==newResume1)); //false

System.*out*.println("小学学历地址是否相同"+(oldResume.getEduExp()==litleEduExperience)); //true

从上述打印结果发现，虽然Resume对象被成功赋值了，但是EduExperience

对象并没有被复制，实现了浅克隆。

（2）如何修改前面代码，实现另一种克隆效果

为了能够在复制Resume的同时也能够复制EduExperience，使用Java中的序列化Serialization的方式实现深克隆，让Resume和EduExperience都实现Serialization接口，代码修改如下：

import java.io.\*;

public class Resume implements Cloneable, Serializable {

private String name;

private String sex;

private EduExperience eduExp;

public Resume(String name, String sex, EduExperience eduExp) {

this.name = name;

this.sex = sex;

this.eduExp = eduExp;

}

// 使用序列化和反序列化进行深克隆

@Override

public Resume clone() {

try {

// 将当前对象序列化到流中

ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();

ObjectOutputStream objectOutputStream = new ObjectOutputStream(byteArrayOutputStream);

objectOutputStream.writeObject(this);

objectOutputStream.flush();

// 从流中反序列化出一个新的对象

ByteArrayInputStream byteArrayInputStream = new ByteArrayInputStream(byteArrayOutputStream.toByteArray());

ObjectInputStream objectInputStream = new ObjectInputStream(byteArrayInputStream);

return (Resume) objectInputStream.readObject();

} catch (IOException | ClassNotFoundException e) {

e.printStackTrace();

}

return null;

}

@Override

public String toString() {

return "基本信息[姓名：" + name + " 性别：" + sex + "] "

+ "学历[学习时间：" + eduExp.getEduTime() + " 学校：" + eduExp.getSchool() + "]";

}

// Getters and Setters

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getSex() {

return sex;

}

public void setSex(String sex) {

this.sex = sex;

}

public EduExperience getEduExp() {

return eduExp;

}

public void setEduExp(EduExperience eduExp) {

this.eduExp = eduExp;

}

}

import java.io.Serializable;

public class EduExperience implements Serializable {

private String school;

private String eduTime;

public EduExperience(String school, String eduTime) {

this.school = school;

this.eduTime = eduTime;

}

@Override

public String toString() {

return "EduExperience{" +

"school='" + school + '\'' +

", eduTime='" + eduTime + '\'' +

'}';

}

// Getters and Setters

public String getSchool() {

return school;

}

public void setSchool(String school) {

this.school = school;

}

public String getEduTime() {

return eduTime;

}

public void setEduTime(String eduTime) {

this.eduTime = eduTime;

}

}

**【总结】**

|  |  |
| --- | --- |
| 实验总结  （不少于100字） | 通过本次实验，我深入理解了原型模式的定义及其应用。原型模式的核心思想是通过克隆现有对象来创建新的对象，从而避免了对象的重复创建，提高了性能。在实验中，我实现了一个个人简历的深克隆和浅克隆的示例，掌握了原型模式中的 **Prototype** 角色、**ConcretePrototype** 角色，以及如何通过 clone() 方法进行浅克隆操作。通过对比浅克隆和深克隆的实现方式，学习了如何通过 Serializable 来实现深克隆。此外，我还熟悉了原型模式的类图和结构图，了解了它在实际编程中的应用。此实验增强了我对设计模式的理解，尤其是在对象复制和性能优化方面的实践能力。 |
| 学号 | 22154A206 |
| 姓名 | 冯艳芳 |
| 成绩 |  |