《Java》课程实验报告

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 专业名称 | 计算机科学与技术 | 年级 | 2023 | 班级 | 01 |
| 学生姓名 | 颜丙超 | 学号 | 202311000415 | 指导教师 | 梁成 |
| 实验题目 | FAN和point | | | 提交时间 | 2024.11.3 |

1. 实验目的和要求

**实验目的**：实验一：1.掌握面向对象编程的基本概念，包括类、对象、数据域、方法、构造方法等。

2.学会如何设计一个类来表示现实世界中的实体，如本题中的风扇（Fan）。

3.理解并实践封装的概念，通过私有数据域和访问器/修改器来保护数据。

4.掌握如何使用构造方法来初始化对象的状态。

5.学会如何编写测试程序来验证类的功能。

实验二：1.掌握面向对象编程基础：通过设计和实现一个名为Point的类，深入理解面向对象编程中的类、对象、属性、方法等基本概念。

2.学习数据结构与算法：通过实现找出距离最近的两个点和计算所有点到最远点的距离的功能，学习和实践相关的数据结构与算法，如距离计算、数组遍历、排序等。

3.提升编程实践能力：通过编写代码、调试程序、处理错误等过程，提升编程实践能力和解决问题的能力。

**实验要求：**实验一：**1.**设计一个名为Fan的类，该类应包含题目中列出的所有常量、数据域、访问器和修改器、构造方法以及toString方法。

2.常量SLOW、MEDIUM和FAST应分别被赋予值1、2和3，表示风扇的不同速度。

3.数据域speed的默认值为SLOW，on的默认值为false，radius的默认值为5.0，color的默认值为"blue"。

4.编写访问器和修改器来访问和修改这些私有数据域的值。

5.编写一个无参构造方法来创建默认状态的风扇对象。

6.编写toString方法来返回描述风扇状态的字符串，根据风扇是否打开来返回不同的信息。

7.编写一个测试程序，创建两个Fan对象。第一个对象为默认状态，第二个对象的特征由用户输入。通过调用它们的toString方法来显示这些对象的状态。

8.测试程序的输出格式应与题目中给出的输出格式提示一致，包括用户输入提示和风扇状态的输出。

实验二：1.：设计Point类

创建一个名为Point的类，该类应包含两个私有属性：x和y，分别表示点的横坐标和纵坐标。

为Point类编写构造方法，用于初始化点的坐标。

编写getter和setter方法，用于访问和修改点的坐标。

编写一个方法，用于计算两个Point对象之间的距离。

2.创建Point对象数组

创建一个Point对象数组，数组的大小M由用户输入确定。

使用Random类生成M个点的随机坐标，并初始化Point对象数组。

3.实现找出距离最近的两个点的函数

遍历Point对象数组，计算每对点之间的距离。

记录并更新最小距离及其对应的两个点。

输出距离最近的两个点的坐标和它们之间的距离。

4.实现计算所有点到最远点的距离的函数

遍历Point对象数组，计算每个点到其他所有点的距离。

记录并更新最大距离及其对应的点（最远点）。

再次遍历Point对象数组，计算每个点到最远点的距离，并输出这些距离。

5.编写测试代码

编写一个主函数，用于测试上述功能。

在主函数中，首先创建Point对象数组并初始化。

调用找出距离最近的两个点的函数，并输出结果。

调用计算所有点到最远点的距离的函数，并输出结果。

6.调试和优化

运行测试代码，观察输出结果是否正确。

如果发现错误或异常，使用调试工具进行调试，并修改代码以修复问题。

优化代码，提高代码的可读性和性能。

1. 实验环境

Windows系统，eclpise软件

三、实验内容及实施

实现内容：实验一：设计Fan类：

1.定义一个名为Fan的类。

2.在类中声明三个常量：SLOW、MEDIUM、FAST，分别表示风扇的低速、中速和高速。

3.在类中声明私有数据域：speed（表示风扇的速度）、on（表示风扇是否打开）、radius（表示风扇的半径）以及color（表示风扇的颜色，尽管题目未明确提及，但为增加类的实用性，我们在此添加）。

为每个私有数据域编写访问器（getter）和修改器（setter）。

4.编写一个无参构造方法，用于初始化对象的状态为默认值。

5.编写一个toString方法，用于返回描述风扇状态的字符串。

编写测试程序：

1.创建一个Fan对象，其状态为默认值。

2.通过用户输入创建一个具有特定状态的Fan对象。

3.调用这两个对象的toString方法，并打印输出它们的状态。

实验二：1.设计并实现一个名为Point的类，该类应包含以下功能：

私有属性：x和y，分别表示点的横坐标和纵坐标。

构造方法：用于初始化点的坐标。

Getter和Setter方法：用于访问和修改点的坐标。

距离计算方法：计算两个Point对象之间的距离。

2.利用Point类，实现以下功能：

创建一个包含M个点的Point对象数组，M的值由用户输入确定。使用Random类生成这些点的随机坐标。

编写一个函数，找出数组中距离最近的两个点，并输出它们的坐标和距离。

编写另一个函数，计算数组中所有点到最远点的距离，并输出这些距离。

实施步骤：

实验一：

1.设计Fan类：

使用IDE（如Eclipse、IntelliJ IDEA或VS Code）创建一个新的Java项目。

在项目中创建一个名为Fan的类文件。

在Fan类中声明常量、数据域、访问器、修改器、构造方法和toString方法。

2.实现Fan类：

为常量赋值：public static final int SLOW = 1;、public static final int MEDIUM = 2;、public static final int FAST = 3;。

为数据域声明变量，并设置默认值（如果需要）。

编写访问器和修改器方法，以允许外部代码访问和修改私有数据域的值。

编写无参构造方法，设置数据域的默认值。

编写toString方法，根据风扇的状态返回相应的字符串。

3.编写测试程序：

在项目中创建一个名为TestFan的主类文件。

在TestFan类中编写main方法。

在main方法中创建两个Fan对象：一个使用无参构造方法创建，另一个通过用户输入创建。

使用Scanner类从控制台读取用户输入，并根据输入设置第二个Fan对象的状态。

调用这两个对象的toString方法，并打印输出它们的状态。

4.运行和测试：

在IDE中运行TestFan类的main方法。

观察控制台输出，确保两个Fan对象的状态正确显示。

根据需要调整代码，以修复任何错误或改进功能。

实验二：1.：设计Point类

创建一个名为Point的类，该类应包含两个私有属性：x和y，分别表示点的横坐标和纵坐标。

为Point类编写构造方法，用于初始化点的坐标。

编写getter和setter方法，用于访问和修改点的坐标。

编写一个方法，用于计算两个Point对象之间的距离。

2.创建Point对象数组

创建一个Point对象数组，数组的大小M由用户输入确定。

使用Random类生成M个点的随机坐标，并初始化Point对象数组。

3.实现找出距离最近的两个点的函数

遍历Point对象数组，计算每对点之间的距离。

记录并更新最小距离及其对应的两个点。

输出距离最近的两个点的坐标和它们之间的距离。

4.实现计算所有点到最远点的距离的函数

遍历Point对象数组，计算每个点到其他所有点的距离。

记录并更新最大距离及其对应的点（最远点）。

再次遍历Point对象数组，计算每个点到最远点的距离，并输出这些距离。

5.编写测试代码

编写一个主函数，用于测试上述功能。

在主函数中，首先创建Point对象数组并初始化。

调用找出距离最近的两个点的函数，并输出结果。

调用计算所有点到最远点的距离的函数，并输出结果。

6.调试和优化

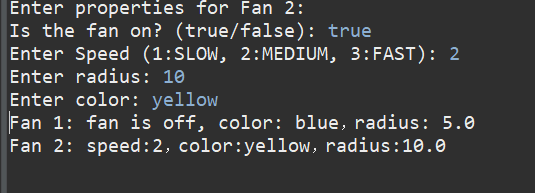
运行测试代码，观察输出结果是否正确。

如果发现错误或异常，使用调试工具进行调试，并修改代码以修复问题。

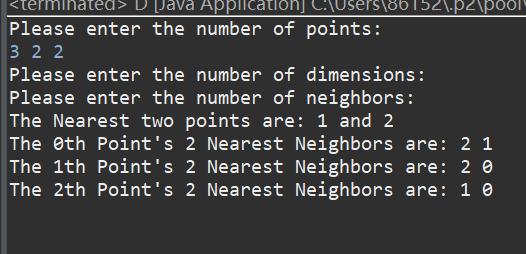
优化代码，提高代码的可读性和性能。

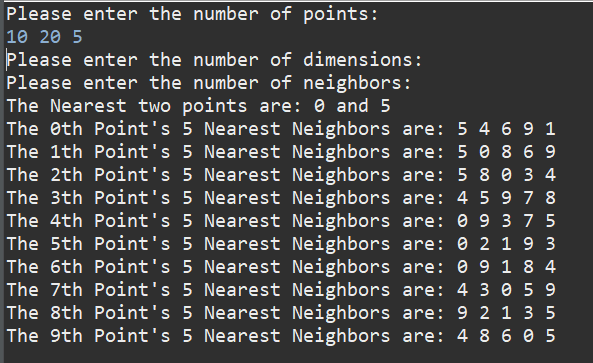
**四、实验结果 (程序的执行结果)**

**实验一：**



实验二：





1. **实验讨论（可选）**
2. 在设计和实现Fan类的过程中，我深刻体会到了面向对象编程（OOP）的核心理念——封装、继承、多态和抽象。通过封装，我们能够将数据和方法组合在一起，形成一个独立的实体（即对象），并通过访问器和修改器来保护数据。这种设计方式不仅提高了代码的可读性和可维护性，还有助于减少错误和提高程序的健壮性**。**
3. 在编写测试程序时，我遇到了如何获取用户输入并设置Fan对象状态的问题。通过查阅Java的Scanner类文档和相关的教程，我学会了如何使用Scanner类从控制台读取用户输入，并将其转换为适当的类型以设置Fan对象的状态。此外，我还学会了如何使用System.out.println方法来打印输出对象的状态，以便验证类的功能是否正确。
4. 加深了对面向对象编程的理解，掌握了类的设计、对象的创建和方法的调用等基本概念。
5. 学习了如何使用数组和Random类来生成和处理随机数据。
6. 掌握了距离计算的基本方法，并能够将其应用于实际问题中。
7. 提高了编程实践能力和解决问题的能力，学会了如何调试和优化代码**。**

**实验一代码：**01 package point;

02

03 import java.util.Scanner;

04

05

06 public class C

07 {

08

09 public static void main(String[] args)

10 {

11

12         Scanner input = new Scanner(System.in);

13

14         Fan f1 = new Fan();

15         Fan f2 = new Fan();

16

17         System.out.println("Enter properties for Fan 2:");

18

19         System.out.print("Is the fan on? (true/false): ");

20

21         boolean isOn = input.nextBoolean();

22

23         f2.setOn(isOn);

24

25

26         System.out.print("Enter speed (1: SLOW, 2: MEDIUM, 3: FAST): ");

27

28         int speed = input.nextInt();

29

30         f2.setSpeed(speed);

31

32

33         System.out.print("Enter radius: ");

34

35         double radius = input.nextDouble();

36

37         f2.setRadius(radius);

38

39

40         System.out.print("Enter color: ");

41

42         String color = input.next();

43

44         f2.setColor(color);

45

46         System.out.println("Fan 1: " + f1);

47

48         System.out.println("Fan 2: " + f2);

49

50

51         input.close();

52

53

54 }

55

56 }

57

58

59 class Fan

60 {

61

62     final int SLOW = 1,  MEDIUM = 2,  FAST = 3;

63

64     private int speed;

65

66     private boolean on;

67

68     private double radius;

69

70     private String color;

71

72

73     public Fan()

74 {

75

76         speed = SLOW;

77

78         on = **false**;

79

80         radius = 5;

81

82         color = "blue";

83

84

85 }

86

87

88     public void setSpeed(int speed)

89 {

90

91         this.speed = speed;

92

93

94 }

95

96

97     public int getSpeed()

98 {

99

100         return speed;

101

102

103 }

104

105

106     public void setOn(boolean status)

107 {

108

109         on = status;

110

111

112 }

113

114

115     public boolean getOn()

116 {

117

118         return on;

119

120

121 }

122

123

124     public void setRadius(double radius)

125 {

126

127         this.radius = radius;

128

129

130 }

131

132

133     public double getRadius()

134 {

135

136         return radius;

137

138

139 }

140

141

142     public void setColor(String color)

143 {

144

145         this.color = color;

146

147

148 }

149

150

151     public String getColor()

152 {

153

154         return color;

155

156

157 }

158

159

160     public String toString()

161 {

162

163         if (on) {

164

165             return "speed: " + speed + ", color: " + color + ", radius: " + radius;

166

167

168 } else

169 {

170

171             return "fan is off, color: " + color + ", radius: " + radius;

172

173

174 }

175

176

177 }

178

179 }

**实验二代码：**01 package fan;

02

03 import java.util.Scanner;

04

05 import java.util.Random;

06

07 public class D

08 {

09

10

11 public static void main(String[] args)

12 {

13

14 System.out.println("Please enter the number of points:");

15

16 Scanner input = new Scanner(System.in);

17

18 int number = input.nextInt();

19

20 Point[] points = new Point[number];

21

22

23 System.out.println("Please enter the number of dimensions:");

24

25 int dim = input.nextInt();

26

27 for(int i = 0; i < points.length; i++)

28 {

29

30 points[i] = new Point(dim,i);

31

32 }

33

34

35

36 System.out.println("Please enter the number of neighbors:");

37

38 int kneigh = input.nextInt();

39

40 input.close();

41

42 findNearestPoints(points);

43

44 double[][] distMat = new double[points.length][points.length];

45

46 distMat = distMatrix(points);

47

48 findKneighbors(kneigh,points,distMat);

49

50

51 }

52

53

54

55 public static double distance(Point x, Point y)

56 {

57

58

59

60 double dis = 0;

61

62 for(int i = 0; i < x.getFeatures().length; i++)

63 {

64

65 dis = dis + Math.pow(x.getFeatures()[i] - y.getFeatures()[i], 2);

66

67 }

68

69 dis = Math.sqrt(dis);

70

71

72

73 return dis;

74

75 }

76

77

78

79 public static double[][] distMatrix(Point[] points)

80 {

81

82 double[][] distMatrix = new double[points.length][points.length];

83

84 for(int i = 0; i < distMatrix.length; i++)

85 {

86

87 for(int j = i + 1; j < distMatrix[i].length; j++)

88 {

89

90 distMatrix[i][j] = distance(points[i], points[j]);

91

92 distMatrix[j][i] = distMatrix[i][j];

93

94 }

95

96 }

97

98 return distMatrix;

99

100 }

101

102

103

104 public static void findNearestPoints(Point[] points)

105

106 {

107

108 double minValue = Double.MAX\_VALUE;

109

110 int minX = -1, minY = -1;

111

112 for(int i = 0; i < points.length - 1; i++)

113 {

114

115 for(int j = i + 1; j < points.length; j++)

116 {

117

118 double temp = distance(points[i], points[j]);

119 if(temp < minValue)

120 {

121

122 minValue = temp;

123

124 minX = i;

125

126 minY = j;

127

128 }

129

130 }

131

132 }

133

134 System.out.print("The Nearest two points are: " + minX + " and " + minY);

135

136 }

137

138

139

140 public static void findKneighbors(int kneigh, Point[] points, double[][] distMatrix)

141 {

142

143

144

145 for(int i = 0; i < points.length; i++)

146 {

147

148

149

150 points[i].neighbors = new Point[kneigh];

151

152 System.out.print("\nThe "+ i +"th Point's "+ kneigh + " Nearest Neighbors are: ");

153

154 boolean[] neighFlag = new boolean[distMatrix[i].length];

155

156 for(int count = 0; count < kneigh; count++)

157 {

158

159

160

161 double minValue = Double.MAX\_VALUE;

162

163

164

165 int pos = -1;

166

167

168

169 for(int j = 0; j < distMatrix[i].length; j++)

170 {

171

172 if(distMatrix[i][j] < minValue && !neighFlag[j]&& i!=j)

173 {

174

175 minValue = distMatrix[i][j];

176

177 pos = j;

178

179 }

180

181 }

182

183 points[i].neighbors[count] = points[pos];

184

185 neighFlag[pos] = **true**;

186

187 System.out.print(pos+" ");

188

189

190

191 }

192

193

194

195 }

196

197 }

198

199 }

200

201

202 class Point

203 {

204

205

206

207 private double[] features;

208

209 Point[] neighbors;

210

211 public Point(int dimension,int seed)

212 {

213

214 features = new double[dimension];

215

216 Random random1 = new Random((seed+500)\*6);

217

218 for(int i = 0; i < features.length; i++)

219 {

220

221 features[i] = random1.nextDouble();

222

223 }

224

225 }

226

227

228

229 public Point(double[] features)

230 {

231

232 this.features = features;

233

234 }

235

236

237

238 public double[] getFeatures()

239 {

240

241 return features;

242

243 }

244

245

246

247 }

248