1 //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 2 //  
 3 // Name: Derek Gallardo  
 4 // Date: July 5th, 2022  
 5 // Version: 1.0.0  
 6 // Programming Language: Java  
 7 // Java Version: 17  
 8 // Description: Create a class for a Major that can be extended for a variety of different college majors using inheritance.  
 9 //  
10 //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
11   
12 import java.io.\*;  
13 import java.util.\*;  
14 import java.util.ArrayList;  
15   
16 public class MajorTester {  
17   
18 public static void main(String[] args) {  
19   
20 // Define & Construct List of Majors  
21 Major m = new Major("Default Major", 2.0, 120);  
22 ComputerScience n = new ComputerScience("Computer Science", 3.0, 120);  
23 ComputerScience o = new DataScience("Data Science", 3.1, 120, false, false);  
24 ComputerScience p = new DataScience("Data Science", 3.1, 120, false, false);  
25 English e = new English("English", 3.2, 120, true, false);  
26   
27 // Print content of Majors  
28 System.out.println(m.toString());  
29 System.out.println(n.toString());  
30 System.out.println(o.toString());  
31 System.out.println(p.toString());  
32 System.out.println(e.toString());  
33 // ((DataScience)o).setIsMajorFull(true);  
34 System.out.println("Do you need to fluent in English to major in English? " + e.getEnglishRequirement());  
35 System.out.println("Is Data Science at full capacity? " + ((DataScience)o).getIsMajorFull());  
36 System.out.println("Do these majors have equal values? " + o.equals(p));  
37   
38 // Create an Array of Major and print the Array List.  
39 ArrayList<Major> majorList = new ArrayList<Major>();  
40 majorList.add(m);  
41 majorList.add(n);  
42 majorList.add(o);  
43 majorList.add(e);  
44 System.out.println(majorList);  
45   
46 }  
47 }

1 import java.util.\*;  
 2   
 3 public class Major {  
 4   
 5 // Private members to be shared among other majors  
 6   
 7 private String majorName;  
 8 private double minimumGPA;  
 9 private int creditsRequired;  
10   
11 // Major Constructors  
12   
13 public Major (String majorName, double minimumGPA, int creditsRequired) {  
14 this.majorName = majorName;  
15 this.minimumGPA = minimumGPA;  
16 this.creditsRequired = creditsRequired;   
17 }  
18   
19 // Getters & Setters   
20   
21 public String getMajorName() {  
22 return majorName;  
23 }  
24   
25 public void setMajorName(String majorName) {  
26 this.majorName = majorName;  
27 }  
28   
29 public double getMinimumGPA() {  
30 return minimumGPA;  
31 }  
32   
33 public void setMinimumGPA(double minimumGPA) {  
34 this.minimumGPA = minimumGPA;  
35 }  
36   
37 public int getCreditsRequired() {  
38 return creditsRequired;  
39 }  
40   
41 public void setCreditsRequired(int creditsRequired) {  
42 this.creditsRequired = creditsRequired;  
43 }  
44   
45 // Override equals method  
46   
47 @Override  
48 public boolean equals(Object m) {  
49   
50 if (m == this) {  
51 return true;  
52 }  
53   
54 if (!(m instanceof Major)) {  
55 return false;  
56 }  
57   
58 // typecast o to Major so that we can compare data members  
59 Major n = (Major) m;  
60   
61 // Compare the data members and return accordingly  
62 return Double.compare(minimumGPA, n.minimumGPA) == 0 && Integer.compare(creditsRequired, n.creditsRequired) == 0;  
63   
64 }  
65   
66 // Override toString method   
67   
68 @Override  
69 public String toString() {  
70 return this.majorName + " " + this.minimumGPA + " " + this.creditsRequired;  
71   
72 }   
73 }

1 import java.io.\*;  
 2   
 3 public class ComputerScience extends Major {  
 4   
 5   
 6 // CS Constructor  
 7 public ComputerScience(String majorName, double minimumGPA, int creditsRequired) {  
 8 super(majorName, minimumGPA, creditsRequired);  
 9   
10 }  
11   
12 // Override Equals Method  
13   
14 @Override  
15 public boolean equals(Object m) {  
16   
17 if (m == this) {  
18 return true;  
19 }  
20   
21 if (!(m instanceof Major)) {  
22 return false;  
23 }  
24   
25 // typecast o to Major so that we can compare data members  
26 Major n = (Major) m;  
27   
28 // Compare the data members and return accordingly  
29 return Double.compare(getMinimumGPA(), n.getMinimumGPA()) == 0 && Integer.compare(getCreditsRequired(), n.getCreditsRequired()) == 0;  
30   
31 }  
32   
33 // Override toSting method  
34 @Override  
35 public String toString() {  
36 return getMajorName() + " " + getMinimumGPA() + " " + getCreditsRequired();  
37   
38 }   
39   
40   
41 }

1 import java.util.\*;  
 2   
 3 public class DataScience extends ComputerScience {  
 4   
 5 // Two Extra Private Members  
 6   
 7 private boolean fullCapacity;  
 8 private boolean csPrerequisitiesRequired;  
 9   
10 // DS Constructor  
11   
12 public DataScience(String majorName, double minimumGPA, int creditsRequired, boolean fullCapacity, boolean csPrerequisitiesRequired) {  
13 super(majorName, minimumGPA, creditsRequired);  
14 this.fullCapacity = fullCapacity;  
15 this.csPrerequisitiesRequired = csPrerequisitiesRequired;  
16 }  
17   
18 // Two extra getters and setters  
19   
20 public boolean getIsMajorFull() {  
21 return fullCapacity;  
22 }  
23   
24 public void setIsMajorFull(boolean fullCapacity) {  
25 this.fullCapacity = fullCapacity;  
26 }  
27   
28 public boolean getCSPrerequisitiesRequired() {  
29 return csPrerequisitiesRequired;  
30 }  
31   
32 public void setCSPrerequisitiesRequirement(boolean csPrerequisitiesRequired) {  
33 this.csPrerequisitiesRequired = csPrerequisitiesRequired;  
34 }  
35   
36 // Override toString method  
37   
38 @Override  
39 public String toString() {  
40 return getMajorName() + " " + getMinimumGPA() + " " + getCreditsRequired() + " " + this.fullCapacity + " " + this.csPrerequisitiesRequired;   
41 }  
42   
43 // Override Equals method  
44   
45 @Override  
46 public boolean equals(Object m) {  
47   
48 if (m == this) {  
49 return true;  
50 }  
51   
52 if (!(m instanceof Major)) {  
53 return false;  
54 }  
55   
56 // typecast o to Major so that we can compare data members  
57 Major n = (Major) m;  
58   
59 // Compare the data members and return accordingly  
60 return Double.compare(getMinimumGPA(), n.getMinimumGPA()) == 0 && Integer.compare(getCreditsRequired(), n.getCreditsRequired()) == 0;  
61   
62 }  
63   
64 }

1 import java.util.\*;  
 2   
 3 public class English extends Major {  
 4   
 5 // Two additional private members  
 6   
 7 private boolean englishProficienyRequired;  
 8 private boolean foreignLanguageRequired;  
 9   
10 // English Major Constructor  
11   
12 public English (String majorName, double minimumGPA, int creditsRequired, boolean englishProficienyRequired, boolean foreignLanguageRequired) {  
13 super(majorName, minimumGPA, creditsRequired);  
14 this.foreignLanguageRequired = foreignLanguageRequired;  
15 this.englishProficienyRequired = englishProficienyRequired;  
16 }  
17   
18 // Two extra getters and setters  
19   
20 public boolean getEnglishRequirement() {  
21 return englishProficienyRequired;  
22 }  
23   
24 public boolean getForeignLanguageRequirement() {  
25 return foreignLanguageRequired;  
26 }  
27   
28 public void setEnglishRequirement (boolean englishProficienyRequired) {  
29 this.englishProficienyRequired = englishProficienyRequired;  
30 }  
31   
32 public void setForeignLanguageRequirement (boolean foreignLanguageRequired) {  
33 this.foreignLanguageRequired = foreignLanguageRequired;  
34 }  
35   
36 // Override toString method  
37   
38 @Override  
39 public String toString() {  
40 return getMajorName() + " " + getMinimumGPA() + " " + getCreditsRequired() + " " + this.englishProficienyRequired + " " + this.foreignLanguageRequired;   
41 }  
42   
43 // OPverride Equals method  
44   
45 @Override  
46 public boolean equals(Object m) {  
47   
48 if (m == this) {  
49 return true;  
50 }  
51   
52 if (!(m instanceof Major)) {  
53 return false;  
54 }  
55   
56 // typecast o to Major so that we can compare data members  
57 Major n = (Major) m;  
58   
59 // Compare the data members and return accordingly  
60 return Double.compare(getMinimumGPA(), n.getMinimumGPA()) == 0 && Integer.compare(getCreditsRequired(), n.getCreditsRequired()) == 0;  
61   
62 }  
63   
64 }