1 /\*\*  
 2 \* The MajorTester class creates testing objects for the Major class and all  
 3 \* of the classes that extend Major:  
 4 \* English (Art), Math (Computer Science), Business (Management, Administration)  
 5 \* The main method creates and prints an array of Major objects.  
 6 \* The updateMajors() class updates the information for each subclass.  
 7 \* The main method then prints again the Major objects with the new information.  
 8 \*   
 9 \* @author Stephanie Gremillion  
 10 \* @version 17.0.2  
 11 \* @since 2022/07/06  
 12 \*/  
 13   
 14 public class MajorTester {  
 15 public static void main(String[] args) {  
 16 // creating objects in array  
 17 Major[] majors = new Major[8];  
 18 majors[0] = new Major("TBD", 2.50, 23);  
 19 majors[1] = new English("English", 2.75, 30, 32);  
 20 majors[2] = new Math("Math", 3.00, 25, 35);  
 21 majors[3] = new Business("Business", 2.75, 25, 1);  
 22 majors[4] = new Art("Art", 2.50, 25, 25, "Art 101", "Portfolio");  
 23 majors[5] = new ComputerScience("Computer Science", 3.00, 30, 32, "Advisor Permission Required", "Math 110");  
 24 majors[6] = new Management("Business Management", 2.75, 27, 2, "English 101");  
 25 majors[7] = new Administration("Business Administration", 3.00, 30, 2, "Stats 110", "French 101 & 102");  
 26   
 27 // setting prereq  
 28 majors[0].setPrereq();  
 29 majors[1].setPrereq();  
 30 majors[2].setPrereq();  
 31 majors[3].setPrereq();  
 32 majors[4].setPrereq();  
 33 majors[5].setPrereq();  
 34 majors[6].setPrereq();  
 35 majors[7].setPrereq();  
 36   
 37 // printing majors information  
 38 for(int ii = 0; ii < 2; ii++) { // prints information twice, second time updated with new information  
 39 for(int i = 0; i < 8; i++) {  
 40 System.out.println("Major: " + majors[i].getName());  
 41 System.out.printf("Minimum GPA: %.2f\n", majors[i].getGPA());  
 42 System.out.println("Minimum ACT: " + majors[i].getACT());  
 43 System.out.println("Core Prerequisite: " + majors[i].getPrereq());  
 44 switch(i) {  
 45 case 1: // prints English information  
 46 System.out.println("Minimum English ACT: " + majors[i].getEngAct());  
 47 System.out.println(majors[i].toString());  
 48 break;  
 49 case 2: // prints Math information  
 50 System.out.println("Minimum Math ACT: " + majors[i].getMathAct());  
 51 System.out.println(majors[i].toString());  
 52 break;  
 53 case 3: // prints Business information  
 54 System.out.println("Class Year Requirement: " + majors[i].getYear());  
 55 System.out.println(majors[i].toString());  
 56 break;  
 57 case 4: // prints Art information  
 58 System.out.println("Minimum English ACT: " + majors[i].getEngAct());  
 59 System.out.println("Other Requirements: " + majors[i].getOtherReq());  
 60 System.out.println(majors[i].toString());  
 61 break;  
 62 case 5: // prints ComputerScience information  
 63 System.out.println("Minimum Math ACT: " + majors[i].getMathAct());  
 64 System.out.println("Requirements: " + majors[i].getPermission());  
 65 System.out.println("Math Prerequisite: " + majors[i].getMathPrereq());  
 66 System.out.println(majors[i].toString());  
 67 break;  
 68 case 6: // prints Management information  
 69 System.out.println("Class Year Requirement: " + majors[i].getYear());  
 70 System.out.println("English Prerequisite: " + majors[i].getEngPrereq());  
 71 System.out.println(majors[i].toString());  
 72 break;  
 73 case 7: // prints Administration information  
 74 System.out.println("Class Year Requirement: " + majors[i].getYear());  
 75 System.out.println("Statistics Prerequisite: " + majors[i].getStatsPrereq());  
 76 System.out.println("Language Requirement: " + majors[i].getLanguage());  
 77 System.out.println(majors[i].toString());  
 78 break;  
 79 default:  
 80 break;  
 81 }  
 82 System.out.println(); // empty line  
 83 }  
 84   
 85 if(ii != 0) {   
 86 System.out.println(majors[1].equals(majors[5].getName(), majors[5].getGPA(), majors[5].getACT())); // prints comparison of English and Computer Science  
 87 System.out.println(); // empty line  
 88 }  
 89   
 90 if(ii == 0) {  
 91 System.out.println(majors[5].equals(majors[5].getName(), majors[5].getGPA(), majors[5].getACT())); // prints comparison of Computer Science and Computer Science  
 92 System.out.println(); // empty line  
 93 System.out.println("------------------------------");  
 94 System.out.println("Updating majors information...");  
 95 System.out.println("------------------------------");  
 96 System.out.println(); // empty line  
 97 updateMajors(majors); // updates the majors with new information  
 98 }  
 99 }  
100 }  
101   
102 // using the setters to update major information  
103 public static void updateMajors(Major[] majors) {  
104 // updating majors information  
105 majors[0].setName("Undeclared"); // setting new Major name  
106   
107 majors[1].setGPA(3.00); // setting new English GPA requirement  
108 majors[1].setACT(27); // setting new English ACT requirement  
109 majors[1].setEngAct(30); // setting new English eng ACT requirement  
110   
111 majors[2].setGPA(2.75); // setting new Math GPA requirement  
112 majors[2].setACT(30); // setting new Math ACT requirement  
113 majors[2].setMathAct(30); // setting new Math math ACT requirement  
114   
115 majors[3].setGPA(2.50); // setting new Business GPA requirement  
116 majors[3].setACT(27); // setting new Business ACT requirement  
117 majors[3].setYear(2); // setting new Business year requirement  
118   
119 majors[4].setGPA(2.75); // setting new Art GPA requirement  
120 majors[4].setACT(27); // setting new Art ACT requirement  
121 majors[4].setEngAct(32); // setting new Art eng ACT requirement  
122 majors[4].setOtherReq("E-Portfolio"); // setting new Art other requirement  
123   
124 majors[5].setGPA(3.75); // setting new Computer Science GPA requirement  
125 majors[5].setACT(32); // setting new Computer Science ACT requirement  
126 majors[5].setMathAct(30); // setting new Computer Science math ACT requirement  
127 majors[5].setMathPrereq("Math 220"); // setting new Computer Science math prerequisite  
128 majors[5].setPermUnreq(); // setting Computer Science advisor permission unrequired  
129   
130 majors[6].setName("Management"); // setting new Management name  
131 majors[6].setGPA(3.00); // setting new Management GPA requirement  
132 majors[6].setACT(30); // setting new Management ACT requirement  
133 majors[6].setCorePrereq("Speech 120"); // setting new Management core prerequisite;  
134 majors[6].setEngPrereq("English 210"); // setting new Managagement english prerequisite  
135 majors[6].setYear(3); // setting new Management year requirement  
136   
137 majors[7].setName("Administration"); // setting new Administration name  
138 majors[7].setGPA(3.25); // setting new Administration GPA requirement  
139 majors[7].setACT(32); // setting new Administration ACT requirement  
140 majors[7].setCorePrereq("Speech 120"); // setting new Administration core prerequisite  
141 majors[7].setStatsPrereq("Stats 210"); // setting new Administration stats prerequisite  
142 majors[7].setLangUnreq(); // setting Administration language requirement unrequired  
143 majors[7].setYear(3); // setting new Administration year requirement  
144 }  
145 }

1 /\*\*  
 2 \* The Major class creates a new major with attributes:  
 3 \* name, GPA, ACT  
 4 \* This class also creates the method prototypes used for the subclasses.  
 5 \* The equals() method in this class overrides the default method.  
 6 \*   
 7 \* @author Stephanie Gremillion  
 8 \* @version 17.0.2  
 9 \* @since 2022/07/06  
 10 \*/  
 11   
 12 public class Major {  
 13 // variables  
 14 private String majorName;  
 15 private double majorGPA;  
 16 private int majorACT;  
 17 private String majorPrereq;  
 18   
 19 // constructor  
 20 public Major(String name, double gPA, int aCT) {  
 21 majorName = name;  
 22 majorGPA = gPA;  
 23 majorACT = aCT;  
 24 }  
 25   
 26 // setter for majorName  
 27 public void setName(String name) {  
 28 majorName = name;  
 29 }  
 30   
 31 // setter for majorGPA  
 32 public void setGPA(double gPA) {  
 33 majorGPA = gPA;  
 34 }  
 35   
 36 // setter for majorACT  
 37 public void setACT(int aCT) {  
 38 majorACT = aCT;  
 39 }  
 40   
 41 // setter for prereq  
 42 public void setPrereq() {  
 43 majorPrereq = "None";  
 44 }  
 45   
 46 // getter for majorName  
 47 public String getName() {  
 48 return majorName;  
 49 }  
 50   
 51 // getter for majorGPA  
 52 public double getGPA() {  
 53 return majorGPA;  
 54 }  
 55   
 56 // getter for majorACT  
 57 public int getACT() {  
 58 return majorACT;  
 59 }  
 60   
 61 // getter for prereq  
 62 public String getPrereq() {  
 63 return majorPrereq;  
 64 }  
 65   
 66 // equals override  
 67 public String equals(String compareName, double compareGPA, int compareACT) {  
 68 String temp = "";  
 69 if((this.getName().compareTo(compareName) == 0) && (Double.compare(this.getGPA(), compareGPA) == 0) && (this.getACT() == compareACT)) {  
 70 temp = (this.getName() + " and " + compareName + " are the same major.");  
 71 }  
 72 else {  
 73 temp = (this.getName() + " and " + compareName + " are not the same major.");  
 74 }  
 75 return temp;  
 76 }  
 77   
 78 // setting method prototypes for subclasses  
 79 public void setEngAct(int x) {};  
 80 public int getEngAct() { return 0; };  
 81 public void setMathAct(int x) {};  
 82 public int getMathAct() { return 0; };  
 83 public void setCorePrereq(String x) {};  
 84 public String getCorePrereq() { return ""; };  
 85 public void setPermission(String x) {};  
 86 public String getPermission() { return ""; };  
 87 public void setMathPrereq(String x) {};  
 88 public String getMathPrereq() { return ""; };  
 89 public void setPermUnreq() {};  
 90 public void setEngPrereq(String x) {};  
 91 public String getEngPrereq() { return ""; };  
 92 public void setStatsPrereq(String x) {};  
 93 public String getStatsPrereq() { return ""; };  
 94 public void setLanguage(String x) {};  
 95 public String getLanguage() { return ""; };  
 96 public void setLangUnreq() {};  
 97 public int getYear() { return 0; };  
 98 public void setYear(int x) {};  
 99 public String getOtherReq() { return ""; };  
100 public void setOtherReq(String x) {};  
101 }

1 /\*\*  
 2 \* The English class extend the Major class and adds attributes:  
 3 \* englishACT  
 4 \* The toString() method in this class overrides the default method.  
 5 \*   
 6 \* @author Stephanie Gremillion  
 7 \* @version 17.0.2  
 8 \* @since 2022/07/06  
 9 \*/  
10   
11 public class English extends Major {  
12 // variables  
13 private int englishACT;  
14   
15 // constructor  
16 public English(String name, double gPA, int aCT, int engACT) {  
17 super(name, gPA, aCT); // sends data back to Major  
18 englishACT = engACT;  
19 }  
20   
21 // setter for englishACT  
22 public void setEngAct(int engACT) {  
23 englishACT = engACT;  
24 }  
25   
26 // getter for engACT  
27 public int getEngAct() {  
28 return englishACT;  
29 }  
30   
31 // toString override  
32 public String toString() {  
33 return(super.getName() + " major requires English ACT score of " + englishACT);  
34 }  
35 }

1 /\*\*  
 2 \* The Math class extend the Major class and adds attributes:  
 3 \* mathACT  
 4 \* The toString() method in this class overrides the default method.  
 5 \*   
 6 \* @author Stephanie Gremillion  
 7 \* @version 17.0.2  
 8 \* @since 2022/07/06  
 9 \*/  
10   
11 public class Math extends Major {  
12 // variables  
13 private int mathACT;  
14   
15 // constructor  
16 public Math(String name, double gPA, int aCT, int matACT) {  
17 super(name, gPA, aCT); // sends data back to Major  
18 mathACT = matACT;  
19 }  
20   
21 // setter for mathACT  
22 public void setMathAct(int matACT) {  
23 mathACT = matACT;  
24 }  
25   
26 // getter for mathACT  
27 public int getMathAct() {  
28 return mathACT;  
29 }  
30   
31 // toString override  
32 public String toString() {  
33 return(super.getName() + " major requires Math ACT score of " + mathACT);  
34 }  
35 }

1 /\*\*  
 2 \* The Business class extend the Major class and adds attributes:  
 3 \* corePrereq  
 4 \* The toString() method in this class overrides the default method.  
 5 \*   
 6 \* @author Stephanie Gremillion  
 7 \* @version 17.0.2  
 8 \* @since 2022/07/06  
 9 \*/  
10   
11 public class Business extends Major {  
12 // variables  
13 private String corePrereq;  
14 private int year;  
15   
16 // constructor  
17 public Business(String name, double gPA, int aCT, int classYear) {  
18 super(name, gPA, aCT); // sends data back to Major  
19 year = classYear;  
20 }  
21   
22 // setter for corePrereq  
23 public void setPrereq() {  
24 corePrereq = "Speech 101";  
25 }  
26   
27 // setter for year  
28 public void setYear(int classYear) {  
29 year = classYear;  
30 }  
31   
32 // getter for corePrereq  
33 public String getPrereq() {  
34 return corePrereq;  
35 }  
36   
37 // getter for year  
38 public int getYear() {  
39 return year;  
40 }  
41   
42 // toString override  
43 public String toString() {  
44 return(super.getName() + " major requires course prerequisite " + this.getPrereq());  
45 }  
46 }

1 /\*\*  
 2 \* The Art class extend the English class and adds attributes:  
 3 \* prereq  
 4 \* The toString() method in this class overrides the default method.  
 5 \*   
 6 \* @author Stephanie Gremillion  
 7 \* @version 17.0.2  
 8 \* @since 2022/07/06  
 9 \*/  
10   
11 public class Art extends English {  
12 // variables  
13 private String prereq;  
14 private String otherReq;  
15   
16 // constructor  
17 public Art(String name, double gPA, int aCT, int engACT, String prerequisite, String other) {  
18 super(name, gPA, aCT, engACT); // sends data back to English, which sends back to Major  
19 otherReq = other;  
20 }  
21   
22 // setter for prereq  
23 public void setPrereq() {  
24 prereq = "Art 101";  
25 }  
26   
27 // setter for otherReq  
28 public void setOtherReq(String other) {  
29 otherReq = other;  
30 }  
31   
32 // getter for prereq  
33 public String getPrereq() {  
34 return prereq;  
35 }  
36   
37 // getter for otherReq  
38 public String getOtherReq() {  
39 return otherReq;  
40 }  
41   
42 // toString override  
43 public String toString() {  
44 return(super.getName() + " major requires course prerequisite " + this.getPrereq());  
45 }  
46 }

1 /\*\*  
 2 \* The ComputerScience class extend the Math class and adds attributes:  
 3 \* permission, mathPrereq  
 4 \* The toString() method in this class overrides the default method.  
 5 \* The setPermUnreq() method in this class sets "permission" variable to "Advisor Permission Unrequired"  
 6 \*   
 7 \* @author Stephanie Gremillion  
 8 \* @version 17.0.2  
 9 \* @since 2022/07/06  
10 \*/  
11   
12 public class ComputerScience extends Math {  
13 // variables  
14 private String permission;  
15 private String mathPrereq;  
16   
17 // constructor  
18 public ComputerScience(String name, double gPA, int aCT, int matACT, String perm, String mathPre) {  
19 super(name, gPA, aCT, matACT); // sends data back to Math, which sends back to Major  
20 permission = perm;  
21 mathPrereq = mathPre;  
22 }  
23   
24 // setter for permission  
25 public void setPermission(String perm) {  
26 permission = perm;  
27 }  
28   
29 // setter for mathPrereq  
30 public void setMathPrereq(String mathPre) {  
31 mathPrereq = mathPre;  
32 }  
33   
34 // getter for permission  
35 public String getPermission() {  
36 return permission;  
37 }  
38   
39 // getter for mathPrereq  
40 public String getMathPrereq() {  
41 return mathPrereq;  
42 }  
43   
44 // setting permission unrequired  
45 public void setPermUnreq() {  
46 permission = "Advisor Permission Unrequired";  
47 }  
48   
49 // toString override  
50 public String toString() {  
51 return(super.getName() + " major requirements: " + permission + " and " + mathPrereq );  
52 }  
53 }

1 /\*\*  
 2 \* The Management class extend the Business class and adds attributes:  
 3 \* engPrereq  
 4 \* The toString() method in this class overrides the default method.  
 5 \*   
 6 \* @author Stephanie Gremillion  
 7 \* @version 17.0.2  
 8 \* @since 2022/07/06  
 9 \*/  
10   
11 public class Management extends Business {  
12 // variables  
13 private String engPrereq;  
14   
15 // constructor  
16 public Management(String name, double gPA, int aCT, int year, String eng) {  
17 super(name, gPA, aCT, year); // sends data back to Business, which sends back to Major  
18 engPrereq = eng;  
19 }  
20   
21 // setter for engPrereq  
22 public void setEngPrereq(String eng) {  
23 engPrereq = eng;  
24 }  
25   
26 // getter for engPrereq  
27 public String getEngPrereq() {  
28 return engPrereq;  
29 }  
30   
31 // toString override  
32 public String toString() {  
33 return( super.getName() + " major requires course prerequisite " + engPrereq);  
34 }  
35 }

1 /\*\*  
 2 \* The Administration class extend the Business class and adds attributes:  
 3 \* statsPrereq, language  
 4 \* The toString() method in this class overrides the default method.  
 5 \* The setLangUnreq() method in this class sets "language" variable to "Language Course(s) Unrequired"  
 6 \*   
 7 \* @author Stephanie Gremillion  
 8 \* @version 17.0.2  
 9 \* @since 2022/07/06  
10 \*/  
11   
12 public class Administration extends Business {  
13 // variables  
14 private String statsPrereq;  
15 private String language;  
16   
17 // constructor  
18 public Administration(String name, double gPA, int aCT, int year, String stats, String lang) {  
19 super(name, gPA, aCT, year); // sends data back to Business, which sends back to Major  
20 statsPrereq = stats;  
21 language = lang;  
22 }  
23   
24 // setter for statsPrereq  
25 public void setStatsPrereq(String stats) {  
26 statsPrereq = stats;  
27 }  
28   
29 // setter for language  
30 public void setLanguage(String lang) {  
31 language = lang;  
32 }  
33   
34 // getter for statsPrereq  
35 public String getStatsPrereq() {  
36 return statsPrereq;  
37 }  
38   
39 // getter for language  
40 public String getLanguage() {  
41 return language;  
42 }  
43   
44 // setting language unrequired  
45 public void setLangUnreq() {  
46 language = "Language Course(s) Unrequired";  
47 }  
48   
49 // toString override  
50 public String toString() {  
51 return(super.getName() + " major requirements: " + statsPrereq + " and " + language);  
52 }  
53 }