/\*\*

\*

\* @author aaronyang

\*/

import java.text.DecimalFormat;

public class Quarterback {

public static void main( String [] args ){

// create int arrays to save the statistics of the player

int [] yards = { 298, 363, 285, 256, 254, 295, 251, 397, 303, 280, 230, 269, 224, 246, 206, 334 };

int [] attempts = { 34, 46, 36, 27, 42, 44, 38, 45, 45, 46, 31, 45, 34, 35, 30, 36 };

int [] completions = { 23, 30, 24, 19, 24, 30, 22, 37, 30, 25, 22, 30, 22, 22, 23, 24 };

int [] yardsPerAttempt = new int[yards.length];

// define variables

final int HIGHER\_YARDS = 275;

int totalYards = 0;

int totalHigherYards = 0;

int maxYards = 0;

// create DecimalFormat object

DecimalFormat percent = new DecimalFormat( "0.0%");

// create for loop

for ( int i = 0; i < yards.length; i++ ){

totalYards += yards[i]; // calculate the total yards

if( yards[i] > HIGHER\_YARDS )

totalHigherYards ++; // calculate the total number of yards that is larger than 275

yardsPerAttempt[i] = yards[i] / attempts[i]; // calculate yards per attempt

if( maxYards < yards[i] )

maxYards = yards[i]; // find the largest yard

}

// print the results:

System.out.println( "\nAverage number of yards per attempt: ");

for ( int i = 0; i < yards.length; i++ ){

System.out.print( yardsPerAttempt[i] + "\t");

}

System.out.println( "\n\nAverage completion percentage for each of the games: ");

for ( int i = 0; i < yards.length; i++ ){

System.out.print( percent.format((double) completions[i] / (double)attempts[i] ) + "\t");

}

System.out.println( "\n\nTotal yards throws in the entire season: " + totalYards

+ "\nNumber of games where the number of yards was over 275: " + totalHigherYards

+ "\nHighest yards thrown in a game: " + maxYards );

}

}

run:

Average number of yards per attempt:

8 7 7 9 6 6 6 8 6 6 7 5 6 7 6 9

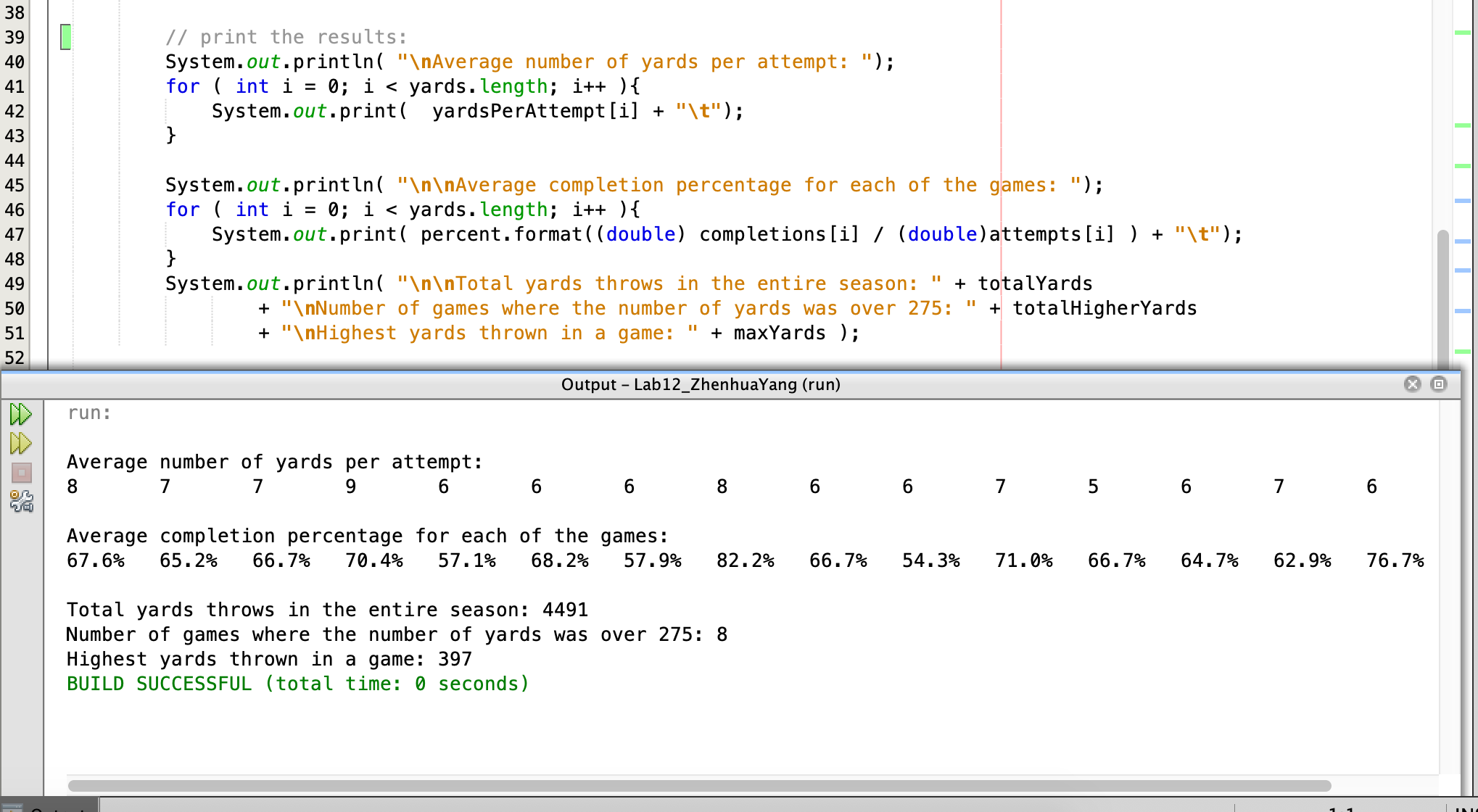
Average completion percentage for each of the games:

67.6% 65.2% 66.7% 70.4% 57.1% 68.2% 57.9% 82.2% 66.7% 54.3% 71.0% 66.7% 64.7% 62.9% 76.7% 66.7%

Total yards throws in the entire season: 4491

Number of games where the number of yards was over 275: 8

Highest yards thrown in a game: 397

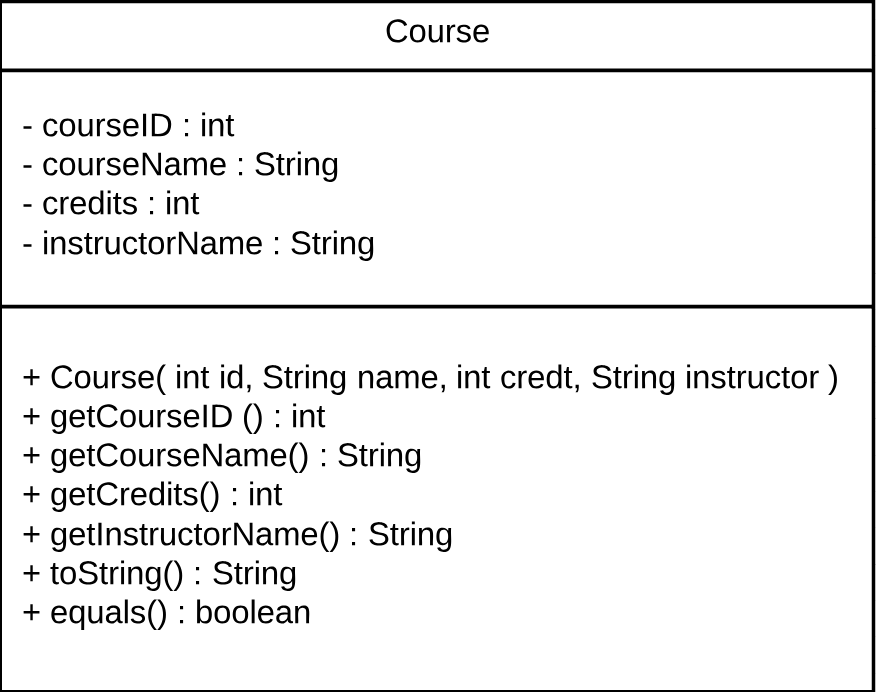


/\*\*

\*

\* @author aaronyang

\*/



public class Course {

// Create instence variables

private int courseID;

private String courseName;

private int credits;

private String instructorName;

/\*\*

\* overloaded constructor

\* @param id is new course ID

\* @param name is new course name

\* @param credt is credits of the course

\* @param instructor is the name of the instructor

\*/

public Course( int id, String name, int credt, String instructor ){

courseID = id;

courseName = name;

credits = credt;

instructorName = instructor;

}

// accessor methods

/\*\*

\*

\* @return course ID

\*/

public int getCourseID(){

return courseID;

}/\*\*

\*

\* @return courseName

\*/

public String getCourseName(){

return courseName;

}

/\*\*

\*

\* @return credits

\*/

public int getGredits(){

return credits;

}

/\*\*

\*

\* @return instructor name

\*/

public String getInstructorName(){

return instructorName;

}

@Override

public String toString(){

return "\nCourse ID: " + courseID

+ "\nCourse Name: " + courseName

+ "\nCredits: " + credits

+ "\nInstructor Name: " + instructorName;

}

@Override

public boolean equals( Object o ){

if ( ! ( o instanceof Course))

return false;

else{

Course objCourse = ( Course )o;

return courseID == objCourse.courseID && courseName == objCourse.courseName

&& credits == objCourse.credits && instructorName == objCourse.instructorName;

}

}

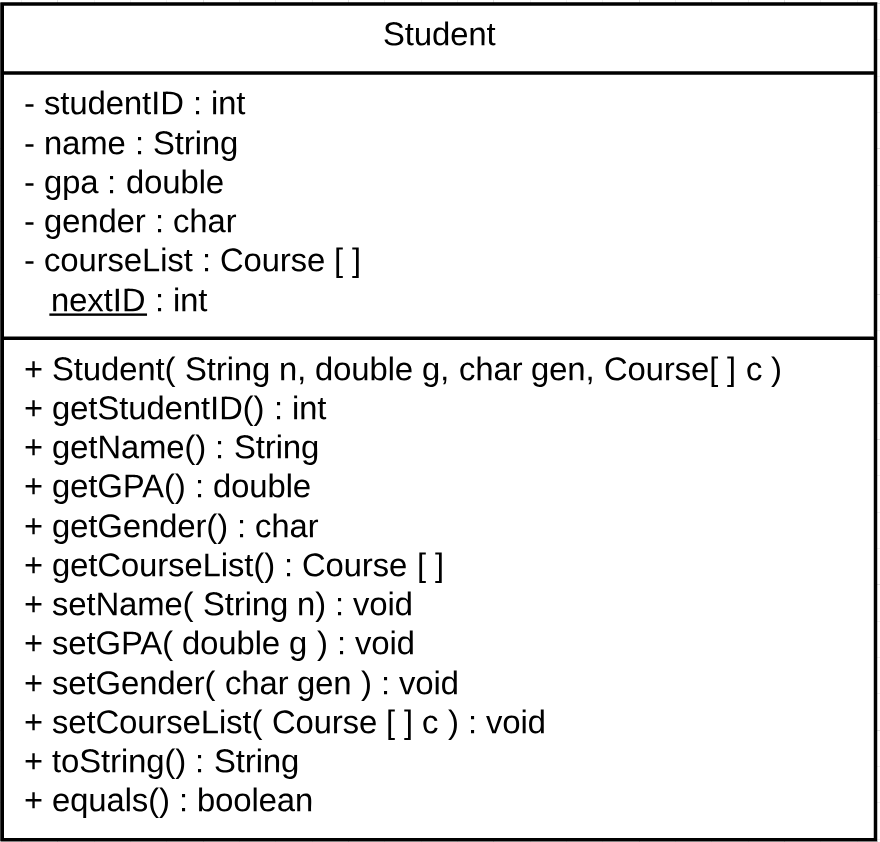
}

/\*\*

\*

\* @author aaronyang

\*/



public class Student {

// instence variables

private int studentID = nextID;

private String name;

private double gpa;

private char gender;

private Course[] courseList;

static int nextID = 1000;

/\*\*

\* overloaded constructor

\* @param n is the String will be assigned to name

\* @param g is the double will be assigned to gpa

\* @param gen is the char will be assigned to gender

\* @param c is the array of object Course to be assigned to courseList

\*/

public Student( String n, double g, char gen, Course[] c ){

name = n;

gpa = g;

gender = gen;

courseList = c;

nextID ++;

}

// accessor methods

/\*\*

\*

\* @return student ID as int

\*/

public int getStudentID(){

return studentID;

}

/\*\*

\*

\* @return Sting name

\*/

public String getName(){

return name;

}

/\*\*

\*

\* @return double gpa

\*/

public double getGPA(){

return gpa;

}

/\*\*

\*

\* @return char gender

\*/

public char getGender(){

return gender;

}

/\*\*

\*

\* @return courseList

\*/

public Course[] getCourseList(){

return courseList;

}

// mutator methods

/\*\*

\*

\* @param n is the String to be assigned to name

\*/

public void setName( String n ){

name = n;

}

/\*\*

\*

\* @param g is double to be assigned to gpa

\*/

public void setGPA( double g ){

gpa = g;

}

/\*\*

\*

\* @param gen is char to be assigned to gender

\*/

public void setGender( char gen ){

gender = gen;

}

/\*\*

\*

\* @param c is the Course array to be assigned to courseList

\*/

public void setCourseList( Course[] c ){

courseList = c;

}

@Override

public String toString(){

String courses = "";

for( int i = 0; i < courseList.length; i++ ){

courses += courseList[i].getCourseName() + "\n";

}

return "========Student Information======\n" + "\n-ID: " + studentID

+ "\n-Name: " + name

+ "\n-GPA: " + gpa

+ "\n-Gener: " + gender

+ "\n-Course List:\n" + courses;

}

/\*\*

\*

\* @param o

\* @return

\*/

@Override

public boolean equals( Object o ){

if( !( o instanceof Student ))

return false;

else{

Student objStudent = (Student)o;

return name == objStudent.name && gpa == objStudent.gpa

&& gender == objStudent.gender

&& courseList == objStudent.courseList;

}

}

}

/\*\*

\*

\* @author aaronyang

\*/

public class StudentsCient {

public static void main( String [] args ){

// create a Course array and define five Course objects inside it.

Course [] courses ={ new Course( 713, "Software Development", 3, "Magel"),

new Course( 715, "Requirement Engineering", 3, "Walia"),

new Course( 887, "Research", 3, "Jun" ),

new Course( 160, "Computer Science 1", 0, "Kotala"),

new Course( 161, "Computer Science 2", 0, "Joe") };

// create two student objects

Student aaronYang = new Student( "Aaron Yang", 3.86, 'M', courses);

Student zhenhuaYang = new Student( "Aaron Yang", 3.86, 'M', courses);

// print the information of two Student objects

System.out.println( aaronYang.toString() );

System.out.println( zhenhuaYang.toString() );

System.out.println( "Are Aaron and Zhenhua the same person (ignore ID)? " + aaronYang.equals(zhenhuaYang));

// print the information of each course

System.out.println( "\n=======Course Information======" );

for( int i = 0; i < courses.length; i++ ){

System.out.println( courses[i].toString() );

}

}

}

run:

========Student Information======

-ID: 1000

-Name: Aaron Yang

-GPA: 3.86

-Gener: M

-Course List:

Software Development

Requirement Engineering

Research

Computer Science 1

Computer Science 2

========Student Information======

-ID: 1001

-Name: Aaron Yang

-GPA: 3.86

-Gener: M

-Course List:

Software Development

Requirement Engineering

Research

Computer Science 1

Computer Science 2

Are Aaron and Zhenhua the same person (ignore ID)? true

=======Course Information======

Course ID: 713

Course Name: Software Development

Credits: 3

Instructor Name: Magel

Course ID: 715

Course Name: Requirement Engineering

Credits: 3

Instructor Name: Walia

Course ID: 887

Course Name: Research

Credits: 3

Instructor Name: Jun

Course ID: 160

Course Name: Computer Science 1

Credits: 0

Instructor Name: Kotala

Course ID: 161

Course Name: Computer Science 2

Credits: 0

Instructor Name: Joe

**Class Course**

* java.lang.Object
  + Course

public class **Course**

extends java.lang.Object

* + ***Constructor Summary***

|  |
| --- |
| **Constructors** |
| **Constructor and Description** |
| [**Course**](Course.html#Course-int-java.lang.String-int-java.lang.String-)(int id, java.lang.String name, int credt, java.lang.String instructor)  overloaded constructor |

* + ***Method Summary***

|  |  |
| --- | --- |
| **All Methods**[**Instance Methods**](javascript:show(2);)[**Concrete Methods**](javascript:show(8);) | |
| **Modifier and Type** | **Method and Description** |
| boolean | [**equals**](Course.html#equals-java.lang.Object-)(java.lang.Object o) |
| int | [**getCourseID**](Course.html#getCourseID--)() |
| java.lang.String | [**getCourseName**](Course.html#getCourseName--)() |
| int | [**getGredits**](Course.html#getGredits--)() |
| java.lang.String | [**getInstructorName**](Course.html#getInstructorName--)() |
| java.lang.String | [**toString**](Course.html#toString--)() |

* + - **Methods inherited from class java.lang.Object**

clone, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

* + ***Constructor Detail***
    - **Course**
    - public Course(int id,
    - java.lang.String name,
    - int credt,

java.lang.String instructor)

overloaded constructor

**Parameters:**

id - is new course ID

name - is new course name

credt - is credits of the course

instructor - is the name of the instructor

* + ***Method Detail***
    - **getCourseID**

public int getCourseID()

**Returns:**

course ID

* + - **getCourseName**

public java.lang.String getCourseName()

**Returns:**

courseName

* + - **getGredits**

public int getGredits()

**Returns:**

credits

* + - **getInstructorName**

public java.lang.String getInstructorName()

**Returns:**

instructor name

* + - **toString**

public java.lang.String toString()

**Overrides:**

toString in class java.lang.Object

* + - **equals**

public boolean equals(java.lang.Object o)

**Overrides:**

equals in class java.lang.Object

**Class Student**

* java.lang.Object
  + Student

public class **Student**

extends java.lang.Object

* + ***Constructor Summary***

|  |
| --- |
| **Constructors** |
| **Constructor and Description** |
| [**Student**](Student.html#Student-java.lang.String-double-char-Course:A-)(java.lang.String n, double g, char gen, [**Course**](Course.html)[] c)  overloaded constructor |

* + ***Method Summary***

|  |  |
| --- | --- |
| **All Methods**[**Instance Methods**](javascript:show(2);)[**Concrete Methods**](javascript:show(8);) | |
| **Modifier and Type** | **Method and Description** |
| boolean | [**equals**](Student.html#equals-java.lang.Object-)(java.lang.Object o) |
| [**Course**](Course.html)[] | [**getCourseList**](Student.html#getCourseList--)() |
| char | [**getGender**](Student.html#getGender--)() |
| double | [**getGPA**](Student.html#getGPA--)() |
| java.lang.String | [**getName**](Student.html#getName--)() |
| int | [**getStudentID**](Student.html#getStudentID--)() |
| void | [**setCourseList**](Student.html#setCourseList-Course:A-)([**Course**](Course.html)[] c) |
| void | [**setGender**](Student.html#setGender-char-)(char gen) |
| void | [**setGPA**](Student.html#setGPA-double-)(double g) |
| void | [**setName**](Student.html#setName-java.lang.String-)(java.lang.String n) |
| java.lang.String | [**toString**](Student.html#toString--)() |

* + - **Methods inherited from class java.lang.Object**

clone, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

* + ***Constructor Detail***
    - **Student**
    - public Student(java.lang.String n,
    - double g,
    - char gen,

[Course](Course.html)[] c)

overloaded constructor

**Parameters:**

n - is the String will be assigned to name

g - is the double will be assigned to gpa

gen - is the char will be assigned to gender

c - is the array of object Course to be assigned to courseList

* + ***Method Detail***
    - **getStudentID**

public int getStudentID()

**Returns:**

student ID as int

* + - **getName**

public java.lang.String getName()

**Returns:**

Sting name

* + - **getGPA**

public double getGPA()

**Returns:**

double gpa

* + - **getGender**

public char getGender()

**Returns:**

char gender

* + - **getCourseList**

public [Course](Course.html)[] getCourseList()

**Returns:**

courseList

* + - **setName**

public void setName(java.lang.String n)

**Parameters:**

n - is the String to be assigned to name

* + - **setGPA**

public void setGPA(double g)

**Parameters:**

g - is double to be assigned to gpa

* + - **setGender**

public void setGender(char gen)

**Parameters:**

gen - is char to be assigned to gender

* + - **setCourseList**

public void setCourseList([Course](Course.html)[] c)

**Parameters:**

c - is the Course array to be assigned to courseList

* + - **toString**

public java.lang.String toString()

**Overrides:**

toString in class java.lang.Object

* + - **equals**

public boolean equals(java.lang.Object o)

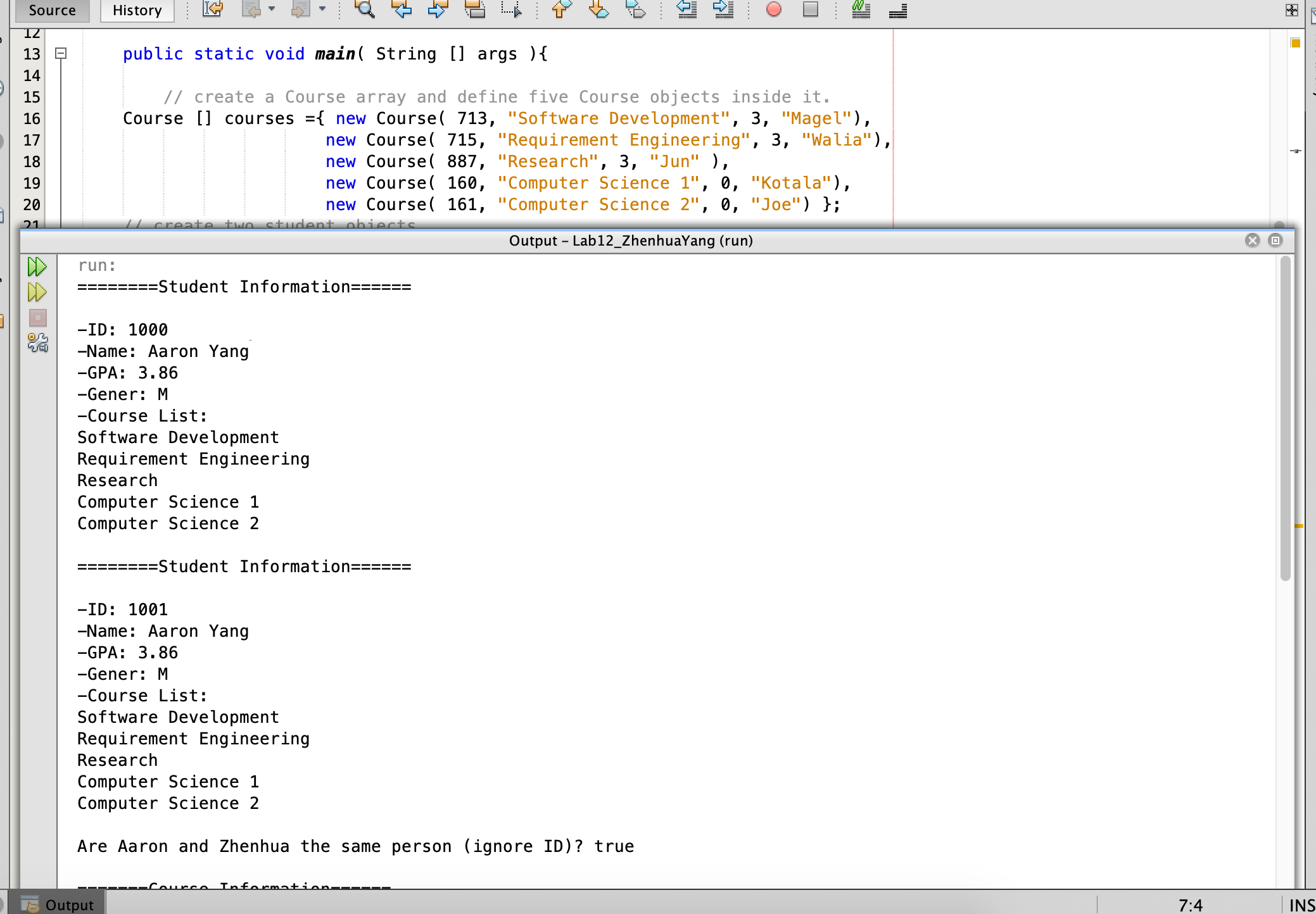
**Overrides:**

equals in class java.lang.Object

**Parameters:**

o -

**Returns:**



/\*\*

\*

\* @author aaronyang

\*/

public class Queue {

private int[] list;

private int count = 0;

/\*\*

\* default constructor of the queue,

\* initializes the list instance variable to an array of length 5

\*/

public Queue( ){

list = new int[5];

}

/\*\*

\* overloaded constructor that takes an int array as a parameter.

\* @param arr

\*/

public Queue( int [] arr ){

list = arr;

count = arr.length;

}

/\*\*

\* accessor method

\* @return count is the size of the current queue.

\*/

public int getCount(){

return count;

}

/\*\*

\* accessor method

\* @return all the elements of the queue in a array.

\*/

public int[] getList(){

return list;

}

/\*\*

\*

\* @param newValue

\*/

public void enqueue( int newValue ){

if( count == list.length ){

addCapacity();

}

list[count] = newValue;

count++;

}

/\*\*

\* remove the first element of the queue

\* @return removed value

\*/

public int dequeue(){

int removed = list[0];

for( int i = 0; i < count; i++ ){

list[i] = list[i+1];

}

count--;

return removed;

}

/\*\*

\*

\* @return max value of the queue

\*/

public int maxValue(){

int max = 0;

for( int i = 0; i < count; i++){

if( max < list[i] )

max = list[i];

}

return max;

}

/\*\*

\* double the size of the queue

\*/

private void addCapacity(){

int [] temp = new int[ list.length \* 2 ];

for( int i = 0; i < count; i++ )

temp[i] = list[i];

list = temp;

}

/\*\*

\*

\* @return all the elements that have been added to the queue

\*/

@Override

public String toString(){

String str = "";

for ( int i = 0; i < count; i++ )

str += list[i] + " ";

return str;

}

/\*\*

\* check if two queues are the same

\* @param l is the queue that is going to be compared.

\* @return true if two queues are the same, or false if two queues are not the same.

\*/

public boolean equals( Queue l ){

if( l.getCount() != count )

return false;

else{

for( int i = 0; i < count; i++ ){

if( l.getList()[i] != list[i] )

return false;

}

return true;

}

}

}

import java.util.Random;

public class QueueTest {

public static void main( String[] args ){

int [] array1 = new int[10]; // create an array variable with the size of 10.

Random ran = new Random(); // create the Random object

// add random numbers into arrqy1

for( int i = 0; i < array1.length; i++ ){

array1[i] = ran.nextInt(100);

}

// create Queue object and assign array1 to it.

Queue queue1 = new Queue(array1);

// print the all the element of queue1.

System.out.println( "Queue1 is: \n" + queue1.toString());

// add five values to queue1 using enqueue method

queue1.enqueue(32);

queue1.enqueue(5);

queue1.enqueue(73);

queue1.enqueue(82);

queue1.enqueue(19);

//print the current queue1.

System.out.println( "\nAfter enqueue five values, Queue1 is: \n" + queue1.toString());

//print the largest value of queue.

System.out.println( "the maximum value of the queue is: " + queue1.maxValue() );

// perform dequeue() method and print the queue1

System.out.println( "\nRemove the first value of Queue1: " + queue1.dequeue());

System.out.println( "After the dequeue operation, Queue1 becomes: \n" + queue1.toString());

// perform dequeue() method again and print the queue1

System.out.println( "\nRemove the first value of Queue1: " + queue1.dequeue());

System.out.println( "After another dequeue operation, Queue1 becomes: \n" + queue1.toString());

// perform dequeue() method again and print the queue1

System.out.println( "\nRemove the first value of Queue1: " + queue1.dequeue());

System.out.println( "After another dequeue operation, Queue1 becomes: \n" + queue1.toString());

}

}

run:

Queue1 is:

4 96 0 2 4 81 39 8 96 77

After enqueue five values, Queue1 is:

4 96 0 2 4 81 39 8 96 77 32 5 73 82 19

the maximum value of the queue is: 96

Remove the first value of Queue1: 4

After the dequeue operation, Queue1 becomes:

96 0 2 4 81 39 8 96 77 32 5 73 82 19

Remove the first value of Queue1: 96

After another dequeue operation, Queue1 becomes:

0 2 4 81 39 8 96 77 32 5 73 82 19

Remove the first value of Queue1: 0

After another dequeue operation, Queue1 becomes:

2 4 81 39 8 96 77 32 5 73 82 19

