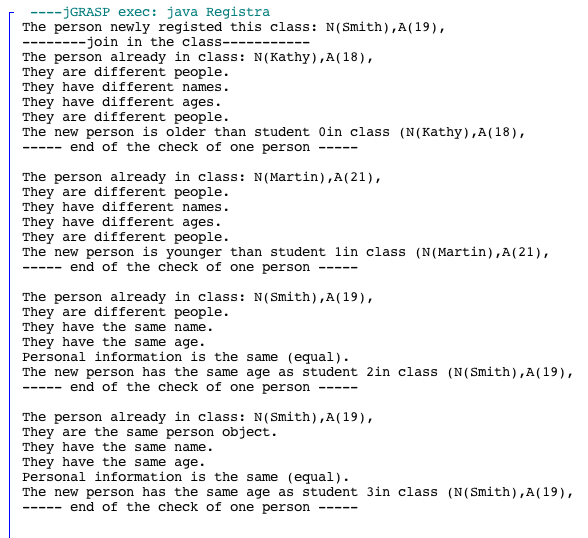
**CSC142, Computer Science II, Project 6 assignment**

Later submission is not accepted.

1. (7 points) Write a class Person with a constructor that accepts a name and an age as its argument. These values should be stored in the *private* attributes “name” and “age”. Then, write necessary accessors and mutators to support the given class Registra as shown in the below. The desired execution is also presented.

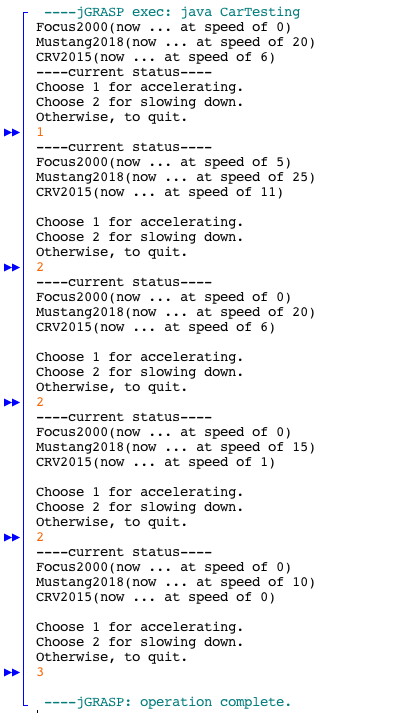
1 public class Registra{  
 2 public static void main(String args[]){  
 3 Person a = new Person("Smith", 19);  
 4 Person [] csc530 = {new Person("Kathy", 18),   
 5 new Person("Martin", 21), new Person("Smith", 19), a};  
 6 System.out.println("The person newly registed this class: "+a);  
 7 System.out.println("--------join in the class-----------");  
 8 for(int i = 0; i<csc530.length; i++){  
 9 System.out.println("The person already in class: "+csc530[i]);  
10   
11 if(a==csc530[i])  
12 System.out.println("They are the same person object.");  
13 else  
14 System.out.println("They are different people.");  
15   
16 if(a.has\_same\_name(csc530[i]))  
17 System.out.println("They have the same name.");  
18 else  
19 System.out.println("They have different names.");  
20   
21 if(a.has\_same\_age(csc530[i]))  
22 System.out.println("They have the same age.");  
23 else  
24 System.out.println("They have different ages.");  
25   
26 if(a.has\_same\_name(csc530[i]) && a.has\_same\_age(csc530[i]))  
27 System.out.println("Personal information is the same (equal).");  
28 else  
29 System.out.println("They are different people.");  
30   
31 if(a.is\_younger\_than(csc530[i]))  
32 System.out.println("The new person is younger than student "+ i   
33 + "in class (" + csc530[i]);  
34 else if (a.is\_older\_than(csc530[i]))  
35 System.out.println("The new person is older than student "+ i   
36 + "in class (" + csc530[i]);  
37 else  
38 System.out.println("The new person has the same age as student "+ i   
39 + "in class (" + csc530[i]);  
40 System.out.println("----- end of the check of one person -----\n");  
41 }   
42 }  
43 }



1. (5 points) Write a class Car with a constructor that accepts a yearModel, a make, and a speed (in MPH) as its argument. For instance, 2000, “Focus,” and 0. These values should be stored in the *private* attributes “yearModel,” “make,” and “speed”. Then, write necessary accessors and mutators to support the given class CarTesting as shown in the below, which include:
   * + accelerate(). This method should add 5 MPH to the speed field each time it is called.
     + brake(). This method should subtract 5 MPH to the speed field each time it is called.

Ensure that the speed field always has a non-negative value.

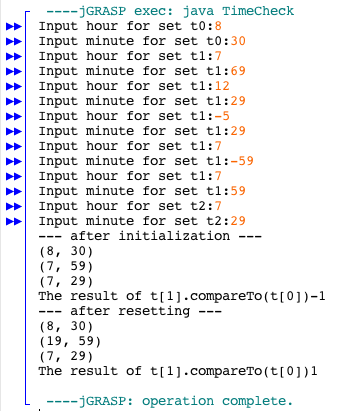
1 import java.util.Scanner;  
 2   
 3 public class CarTesting{  
 4 public static void main(String args[]){  
 5 Scanner kb = new Scanner(System.in);  
 6 Car a = new Car(2000, "Focus", 0);  
 7 Car b = new Car(2018, "Mustang", 20);  
 8 Car c = new Car(2015, "CRV", 6);  
 9 Car [] test\_field = {a, b, c};   
10 for(int i = 0; i<test\_field.length; i++)  
11 System.out.println(test\_field[i]);  
12 System.out.println("----current status----");  
13 System.out.println("Choose 1 for accelerating.");  
14 System.out.println("Choose 2 for slowing down.");  
15 System.out.println("Otherwise, to quit.");  
16 int key = kb.nextInt();  
17 while(key == 1||key==2){  
18 if(key==1){  
19 for(int i = 0; i<test\_field.length; i++){  
20 test\_field[i].accelerate();  
21 }  
22 }  
23 else{  
24 for(int i = 0; i<test\_field.length; i++){  
25 test\_field[i].brake();  
26 }  
27 }  
28 System.out.println("----current status----");  
29 for(int i = 0; i<test\_field.length; i++)  
30 System.out.println(test\_field[i]);  
31 System.out.println("");  
32 System.out.println("Choose 1 for accelerating.");  
33 System.out.println("Choose 2 for slowing down.");  
34 System.out.println("Otherwise, to quit.");  
35 key = kb.nextInt();  
36 }   
37 }  
38 }



1. (7 points) Write a class Time that represents a time of day. It has *private* attributes hour and minute and only these two. The hour value ranges from 0 to 23, where the range 0 to 11 represents a time before noon. The minute value ranges from 0 to 59. The class Time include:
   * + A constructor or initialization to initiate the time (of the new object) to 0 hour and 0 minute.
     + A method “isValid( )” where a call “isValid(hour, minute)” returns true if the given hour and minute values are in the appropriate range.
     + A method “setTime( )” where a call “setTime(hour, minute, isAm)” will set the time attributes. The call will accept the hour from 0 to 11 but will use a 24 hour clock when isAm is false.
     + A method “compareTo( )” to compare two time object and decide which time is earlier, same, or later by its return values -1, 0, and 1 respectively. If needed, develop necessary accessors.

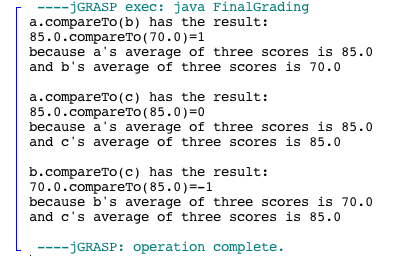
Ensure the support for the following class TimeCheck to get the desired result.

1 import java.util.Scanner;  
 2   
 3 public class TimeCheck{  
 4 public static void main(String args[]){  
 5 Scanner kb = new Scanner(System.in);  
 6 Time [] t = {new Time(), new Time(), new Time()};  
 7 for(int i = 0; i<t.length; i++){  
 8 int hour, minute;  
 9 do{  
10 System.out.print("Input hour for set t"+i+":");  
11 hour = kb.nextInt();  
12 System.out.print("Input minute for set t"+i+":");  
13 minute = kb.nextInt();  
14 }while(!t[i].isValid(hour, minute));  
15 t[i].setTime(hour, minute, true);   
16 }  
17 System.out.println("--- after initialization ---");  
18 for(int i = 0; i<t.length; i++)  
19 System.out.println(t[i]);   
20 int r = t[1].compareTo(t[0]);  
21 System.out.println("The result of t[1].compareTo(t[0])"+r);  
22   
23 t[1].setTime(t[1].getHour(), t[1].getMinute(), false);  
24 System.out.println("--- after resetting ---");  
25 for(int i = 0; i<t.length; i++)  
26 System.out.println(t[i]);   
27 r = t[1].compareTo(t[0]);  
28 System.out.println("The result of t[1].compareTo(t[0])"+r);  
29 }  
30 }



1. (6 points) Write a class TestScores with TWO constructors. One is to accept three scores and save them in the *private* attribute(s). The other is to accept an array with the length >=3 and store the first three elements in these attribute records. Then, write necessary accessors and mutators to support the given class FinalGrading as shown in the below. The desired result is also presented.

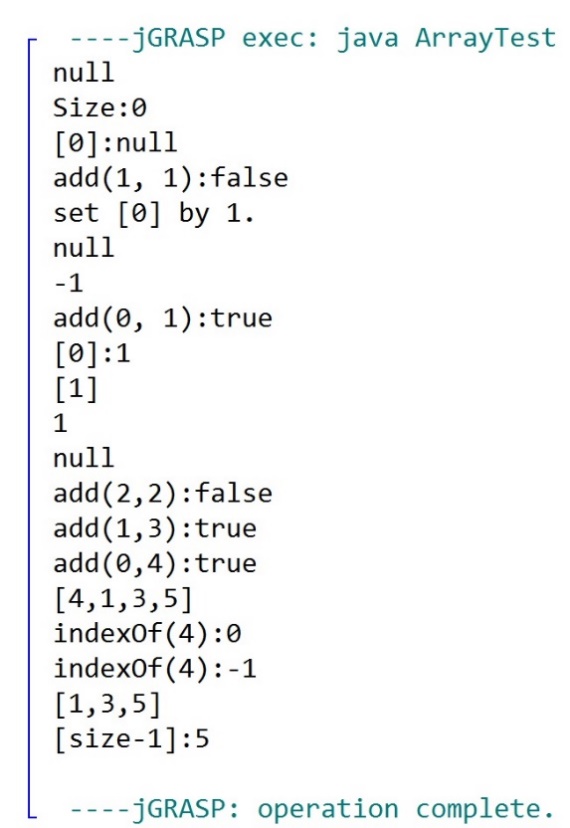
1 public class FinalGrading{  
 2 public static void main(String args[]){  
 3 TestScores a = new TestScores(75, 85, 95);  
 4 int [] t = {65, 70, 75};  
 5 TestScores b = new TestScores(t);  
 6 TestScores c = new TestScores(70, 85, 100);  
 7 // a better solution is [] a = {  
 8 // new T...), new T...(t), new T...}  
 9 int result = a.compareTo(b);  
10 System.out.println("a.compareTo(b) has the result:");  
11 System.out.println(a+".compareTo("+b+")="+result);  
12 System.out.println("because a's average of three scores is "  
13 +a.getAvg());  
14 System.out.println("and b's average of three scores is "  
15 +b.getAvg());  
16 System.out.println("");  
17 result = a.compareTo(c);  
18 System.out.println("a.compareTo(c) has the result:");  
19 System.out.println(a+".compareTo("+c+")="+result);  
20 System.out.println("because a's average of three scores is "  
21 +a.getAvg());  
22 System.out.println("and c's average of three scores is "  
23 +c.getAvg());  
24 System.out.println("");  
25 result = b.compareTo(c);  
26 System.out.println("b.compareTo(c) has the result:");  
27 System.out.println(b+".compareTo("+c+")="+result);  
28 System.out.println("because b's average of three scores is "  
29 +b.getAvg());  
30 System.out.println("and c's average of three scores is "  
31 +c.getAvg());  
32 }  
33 }



1. (10 points) Write the class MyArray with a *private* array attribute “private int [] head” and use this attribute only to support the given class ArrayTest without throwing any exception.

1 public class ArrayTest{  
 2 public static void main(String args[]){  
 3 MyArray a = new MyArray();  
 4 MyArray b = a;  
 5 System.out.println(a);  
 6 a.remove(); // Should not incur any exception.  
 7 System.out.println("Size:"+a.size());  
 8 System.out.println("[0]:"+a.element(0));  
 9 System.out.println("add(1, 1):"+a.add(1,1));  
10 System.out.println("set [0] by 1.");  
11 a.set(0,1); // No exception even there is no element.  
12 System.out.println(a);  
13 System.out.println(a.indexOf(1));  
14 System.out.println("add(0, 1):"+a.add(0,1));  
15 System.out.println("[0]:"+a.element(0));  
16 System.out.println(a);  
17 System.out.println(a.size());  
18 a.clear();  
19 System.out.println(a);  
20 a.add(1);  
21 System.out.println("add(2,2):"+a.add(2, 2));  
22 System.out.println("add(1,3):"+a.add(1, 3));  
23 System.out.println("add(0,4):"+a.add(0, 4));  
24 a.add(5);  
25 System.out.println(a);  
26 System.out.println("indexOf(4):"+a.indexOf(4));  
27 a.remove();  
28 System.out.println("indexOf(4):"+a.indexOf(4));  
29 System.out.println(b); // Not a!  
30 System.out.println("[size-1]:"+b.element(b.size()-1));   
31 }  
32 }

The desired result is listed as follows:



The methods include:

* “add( )” to accept an integer as the argument and insert this value at the end of the current array “head.” The method should also support the insertion when the head is null.
* “toString” to print out all elements as desired, without any unnecessary delimiter “,”.
* “remove( )” that remove the first element in the array head. The method should also support the deletion when the head is null or contains one element only.
* “element( )” that accepts an integer index and returns the index element in the array “head,” in the format of String.
* Overloading method “add” that accepts two integers as the argument. The first indicates the position of the new record inserted into array “head” and the second one indicates the value. The method should avoid any exception when the index becomes out of the appropriate range in according to the head/array length.
* “set” that accepts two integers as the argument. The first indicates the position of the array “head” and the second one indicates the target value for the update.
* “indexOf” that accepts an integer value as the argument. The method must return the location of the first appearance of this value in the array “head.” If not found, return -1. Be aware of the situation when head becomes null.
* “size( )” that returns the size of array “head.”
* “clear” that reset the array “head” null.

Evaluation:

1. Submit required java files to D2L before deadline. Each of them cannot use any import.
2. Each support class cannot use any extra attribute (only use these mentioned in the above specification).
3. 0 will be given if you have any compiling error or execution exception.
4. Each private attribute and instance method will be counted and graded as a whole, and no partial credit will be given at the method level. Total = 6(1)+4(1)+6(1)+5(1)+9(1) = 24 (5) = 35.