Exam

Explanations

Carefully formulate your answers, code, and images

Formulate your answers precise and to the point.

Code shall be written so that it is easy to follow and understand. In some situations suitable comments can contribute to understanding. Small syntactical errors can be tolerated. If some parts of code cannot be exactly produced, it is possible that well formed pseudocode may provide a solution. Do not write more code than necessary; if just a method is requested there is no need to create a whole class. All program code is to be written in Java.

When an array (vector) or an object is drawn, it must be clearly visible by which reference the array or object is referred to, and what data is located inside it. When an array or object contains a reference, the resource that is referred to (an object or array) shall be drawn. All references shall have relevant labels.

Points and grading

In total: 41 points

For grade E at least: 21 points

For grade D at least: 25 points

For grade C at least: 29 points

For grade B at least: 33 points

For grade A at least: 37 points

.

Tasks

Task 1 (3 points + 2 points)

int[][] b = { {1, 2, 3},

{2, 3, 4},

{3, 4, 5} };

int[] u = new int[b[0].length];

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

{

u[i] = b[0][i];

for (int j = 1; j < b.length; j++)

u[i] += b[j][i];

}

int[][] v = new int[3][];

v[0] = b[1];

v[1] = b[0];

v[2] = new int[3];

a) Draw the array referred to by reference u.

b) Draw the array referred to by reference v.

Task 2 (3 points + 3 points + 3 points)

The class Person represents a person:

class Person

{

// the person’s name and year of birth

private String name;

private int birthYear;

public Person (String name, int birthYear)

{

this.name = name;

this.birthYear = birthYear;

}

public String toString ()

{

return "<" + name + ", " + birthYear + ">";

}

public int age ()

{

java.time.LocalDate currentDate = java.time.LocalDate.now ();

int currentYear = currentDate.getYear ();

return currentYear - birthYear;

}

}

a) A static method, averageAge, receives a non-empty array of persons (objects of type Person) and returns the average age for these persons. Create that method.

b) A static method, youngestPerson, receives a non-empty array of persons (objects of type Person) and returns the youngest of these persons. Create that method.

c) Create an array of persons (objects of type Person), and call the methods averageAge and youngestPerson with this array.

Task 3 (3 points + 3 points + 3 points)

The class IntSet manages a set of integers:

public class IntSet

{

// elements in the set

private int[] elements = null;

// assumption: all elements in the argument array are unique

public IntSet (int[] elements)

{

this.elements = new int [elements.length];

for (int pos = 0; pos < elements.length; pos++)

this.elements[pos] = elements[pos];

}

// toString returns a string representation of the set

public String toString ()

{

String s = "{";

if (elements.length == 0)

s = s + "}";

else

{

for (int pos = 0; pos < elements.length - 1; pos++)

s = s + elements[pos] + ", ";

s = s + elements[elements.length - 1] + "}";

}

return s;

}

// contains returns true if the given element

// is a member of the set, and false otherwise.

public boolean contains (int element)

{

// code is missing here

}

// intersection returns a set which contains the elements  
 // in the set that are also members of the given set

public IntSet intersection (IntSet set)

{

// code is missing here

// use the method contains

}

}

Some instances of the class IntSet are created and used like this:

int[] numbers1 = {9, 1, 4, 3, 7, 5};

IntSet set1 = new IntSet (numbers1);

int[] numbers2 = {4, 2, 8, 5, 7};

IntSet set2 = new IntSet (numbers2);

System.out.println (set1);

System.out.println (set2);

IntSet set = set1.intersection (set2);

System.out.println (set);

When this code fragment is executed, the following printout is produced:

{9, 1, 4, 3, 7, 5}

{4, 2, 8, 5, 7}

{4, 7, 5}

a) Implement the method contains.

b) Implement the method intersection.

c) Draw the object referred to by the reference set.

Task 4 (3 point + 3 points + 3 points)

The interface Region defines a two-dimensional area. The classes Circle (defines a circle) and Rectangle (defines a rectangle) implements this interface.

interface Region

{

// perimeter returnes the region’s perimeter

double perimeter ();

// area returnes the region’s area

double area ();

}

class Circle implements Region

{

// the radius of the circle

private double radius;

public Circle (double radius)

{

this.radius = radius;

}

// code is missing here

}

class Rectangle implements Region

{

// the lengths of the sides of the rectangle

private double length;

private double width;

public Rectangle (double length, double width)

{

this.length = length;

this.width = width;

}

// code is missing here

}

a) Make the classes Circle and Rectangle complete: write the missing code.

b) A static method selectRectangles accepts an array of areas of type Region, and returns an array that only contains the areas that are of type Rectangle. Create that method.

c) Create an array that contains both circles (objects of type Circle) and rectangles (objects of type Rectangle). Write code that determines and shows the perimeter and area of these regions. Call the method selectRectangles with the created array as argument.

Task 5 (5 points + 4 points)

A class Association defines an association between a key and a corresponding element. An object of the class can be created and used like this:

Association<Integer, String> a =

new Association<> (new Integer (5), new String ("five"));

Integer k = a.getKey ();

String e = a.getElement ();

System.out.println (a + ": " + k + ", " + e);

When this code fragment is executed, the following printout is obtained:

5 --> five: 5, five

A class Associations manages collections of associations in various ways. It contains a method findElement, that receives a list of associations and a key. The method finds the association in the list that has this key, and returns the corresponding element. You can use the method findElement like this:

java.util.List<Association<Integer, String>> assoc = new java.util.ArrayList<> ();

assoc.add (new Association<Integer, String> (4, "four"));

assoc.add (new Association<Integer, String> (5, "five"));

assoc.add (new Association<Integer, String> (7, "seven"));

Integer key = 7;

String element = Associations.findElement (assoc, key);

System.out.println ((element != null)? element : "element not found");

When this code fragment is executed, the following printout is obtained:

seven

a) Create the class Association.

b) Create the method findElement.