Exam: solution

Tasks: solutions

Task 1 (3 points + 2 points)

a) (3 points)

*12*

*9*

*u*

*6*

b) (2 points)

*4*

*3*

*2*

*v[0]*

*v*

*3*

*2*

*1*

*v[1]*

*v[2]*

*0*

*0*

*0*

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

a) (3 points)

public static double averageAge (Person[] persons)

{

int totalAge = persons[0].age ();

for (int pos = 1; pos < persons.length; pos++)

totalAge += persons[pos].age ();

return (double) totalAge / persons.length;

}

b) (3 points)

public static Person youngestPerson (Person[] persons)

{

Person youngestP = persons[0];

for (int pos = 1; pos < persons.length; pos++)

if (persons[pos].age () < youngestP.age ())

youngestP = persons[pos];

return youngestP;

}

c) (3 points)

Person[] persons = { new Person ("Anders", 1967),

new Person ("Anna", 1996),

new Person ("Sanna", 1984),

new Person ("Ingemar", 1978) };

double averageA = averageAge (persons);

Person youngestP = youngestPerson (persons);

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

a) (3 points)

public boolean contains (int element)

{

boolean isMember = false;

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

if (elements[pos] == element)

{

isMember = true;

break;

}

return isMember;

}

b) (3 points)

public IntSet intersection (IntSet set)

{

int countCommonElements = 0;

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

if (set.contains (elements[pos]))

countCommonElements++;

int[] commonElements = new int[countCommonElements];

int posC = 0;

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

if (set.contains (elements[pos]))

commonElements[posC++] = this.elements[pos];

return new IntSet (commonElements);

}

c) (3 points)

*5*

*7*

*4*

*elements*

*set*

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

a) (3 points)

class Circle implements Region

{

private double radius;

public Circle (double radius)

{

this.radius = radius;

}

public double perimeter ()

{

return 2 \* radius \* Math.PI;

}

public double area ()

{

return radius \* radius \* Math.PI;

}

}

class Rectangle implements Region

{

private double length;

private double width;

public Rectangle (double length, double width)

{

this.length = length;

this.width = width;

}

public double perimeter ()

{

return 2 \* (length + width);

}

public double area ()

{

return length \* width;

}

}

b) (3 points)

public static Rectangle[] selectRectangles (Region[] regions)

{

int countRectangles = 0;

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

if (regions[pos] instanceof Rectangle)

countRectangles++;

Rectangle[] rectangles = new Rectangle[countRectangles];

int posR = 0;

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

if (regions[pos] instanceof Rectangle)

rectangles[posR++] = (Rectangle) regions[pos];

return rectangles;

}

c) (3 points)

Region[] regions = { new Circle (5),

new Circle (4),

new Rectangle (3, 4),

new Circle (7),

new Rectangle (1, 2),

new Rectangle (5, 6) };

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

System.out.println (regions[pos].perimeter () + ", " + regions[pos].area ());

System.out.println ();

Rectangle[] rectangles = selectRectangles (regions);

Task 5 (5 points + 4 points)

a) (5 points)

class Association<K, E>

{

private K key;

private E element;

public Association (K key, E element)

{

this.key = key;

this.element = element;

}

public String toString ()

{

return key + " --> " + element;

}

public K getKey ()

{

return key;

}

public E getElement ()

{

return element;

}

}

b) (4 points)

public static <K, E> E findElement (java.util.List<Association<K, E>> assoc, K key)

{

E element = null;

for (Association<K, E> a : assoc)

if (a.getKey ().equals (key))

{

element = a.getElement ();

break;

}

return element;

}