Tentamen

Förklaringar

Utforma noggrant dina svar, kodavsnitt och bilder

Formulera dina svar kortfattat och noggrant.

Koden ska utformas så att det lätt går att följa och förstå den. I vissa situationer kan lämpliga kommentarer bidra till förståelse. Små syntaktiska fel i koden kan eventuellt tolereras. Om delar i ett kodavsnitt inte kan exakt formuleras, kan möjligen en välutformad pseudokod bidra till lösningen. Man ska inte skriva mer kod än som behövs: om bara en metod krävs, behöver inte en hel klass skapas. All programmeringskod ska skrivas i Java.

När en vektor eller ett objekt ritas, ska det klart framgå vilken referens refererar till denna vektor eller detta objekt, och vilka data som finns inuti denna vektor eller detta objekt. När en vektor eller ett objekt innehåller en referens, ska även den refererade resursen (ett objekt eller en vektor) ritas. Man ska förse alla referenser med relevanta beteckningar.

Antalet poäng och betygsgränser

Totalt: 41 poäng

För betyget E räcker : 21 poäng

För betyget D räcker: 25 poäng

För betyget C räcker: 29 poäng

För betyget B räcker: 33 poäng

För betyget A räcker: 37 poäng

.

Uppgifter

Uppgift 1 (3 poäng + 2 poäng)

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) Rita den vektor som refereras med referensen u.

b) Rita den vektor som refereras med referensen v.

Uppgift 2 (3 poäng + 3 poäng + 3 poäng)

Klassen Person representerar en person:

class Person

{

// personens namn och födelseår

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) En statisk metod, averageAge, tar emot en icke-tom vektor med personer (objekt av typen Person) och returnerar medel ålder för dessa personer. Skapa den metoden.

b) En statisk metod, youngestPerson, tar emot en icke-tom vektor med personer (objekt av typen Person) och returnerar den yngsta av dessa personer. Skapa den metoden

c) Skapa en vektor med personer (objekt av typen Person), och anropa metoderna averageAge och youngestPerson i samband med denna vektor.

Uppgift 3 (3 poäng + 3 poäng + 3 poäng)

Klassen IntSet hanterar en heltalsmängd:

public class IntSet

{

// element i mängden

private int[] elements = null;

// antagande: alla element i argumentvektorn är unika

public IntSet (int[] elements)

{

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

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

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

}

// toString returnerar strängrepresentation av mängden

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 returnerar true om ett givet element finns

// i mängden, annars false.

public boolean contains (int element)

{

// koden saknas här

}

// intersection returnerar en mängd som innehåller de

// element i mängden som även finns i en given mängd

public IntSet intersection (IntSet set)

{

// koden saknas här

// använd metoden contains

}

}

Några instanser av klassen IntSet skapas och används så här:

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);

När detta kodavsnitt exekveras, skapas följande utskrift:

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

{4, 2, 8, 5, 7}

{4, 7, 5}

a) Implementera metoden contains.

b) Implementera metoden intersection.

c) Rita det objekt som refereras med referensen set.

Uppgift 4 (3 poäng + 3 poäng + 3 poäng)

Gränssnittet Region definierar ett tvådimensionellt område. Klasserna Circle (definierar en cirkel) och Rectangle (definierar en rektangel) implementerar detta gränssnitt.

interface Region

{

// perimeter returnerar områdets omkrets

double perimeter ();

// area returnerar områdets area

double area ();

}

class Circle implements Region

{

// cirkelns radie

private double radius;

public Circle (double radius)

{

this.radius = radius;

}

// koden saknas här

}

class Rectangle implements Region

{

// rektangelns sidlängder

private double length;

private double width;

public Rectangle (double length, double width)

{

this.length = length;

this.width = width;

}

// koden saknas här

}

a) Komplettera klasserna Circle och Rectangle: skriv den kod som saknas.

b) En statisk metod selectRectangles tar emot en vektor med områden av typen Region, och returnerar en vektor som endast innehåller de av dessa områden som är av typen Rectangle. Skapa den metoden.

c) Skapa en vektor som innehåller både cirklar (objekt av typen Circle ) och rektanglar (objekt av typen Rectangle). Skriv kod som bestämmer och visar omkrets och area för dessa områden. Anropa metoden selectRectangles med den skapade vektorn som argument.

Uppgift 5 (5 poäng + 4 poäng)

En klass Association definierar en association mellan en nyckel och motsvarande element. Ett objekt av klassen kan skapas och användas så här:

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);

När det här kodavsnittet utförs, erhålls följande utskrift:

5 --> five: 5, five

En klass Associations hanterar samlingar av associationer på olika sätt. Den innehåller en metod findElement, som tar emot en lista med associationer och en nyckel. Metoden hittar den association i listan som har denna nyckel, och returnerar motsvarande element. Man kan använda metoden findElement så här:

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");

När det här kodavsnittet utförs, erhålls följande utskrift:

seven

a) Skapa klassen Association.

b) Skapa metoden findElement.