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: 42 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 (2 poäng + 4 poäng)

public static void main (String[] args)

{

int[][] b = { {1}, {2, 3}, {4, 5, 6} };

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

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

u[i] = b[i][b[i].length - 1];

int[][] v = transform (b);

}

public static int[][] transform (int[][] a)

{

int[][] w = new int[a.length][];

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

w[i] = a[a.length - 1 - i];

return w;

}

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 Rectangle representerar en rektangel:

class Rectangle

{

// rektangelns sidlängder

private double length;

private double width;

// rektangelns färg

private String color;

public Rectangle (double length, double width, String color)

{

this.length = length;

this.width = width;

this.color = color;

}

public String getColor ()

{

return color;

}

public double area ()

{

return length \* width;

}

}

a) En statisk metod, minRectangle, tar emot en vektor med rektanglar (objekt av typen Rectangle) och returnerar den rektangel som har minst area. Skapa den metoden.

b) En statisk metod, selectRectangles, tar emot en vektor med rektanglar (objekt av typen Rectangle) och en färg (ett objekt av typen String), och returnerar de rektanglar (som en vektor) som är av givna färgen. Skapa den metoden.

c) Skapa en vektor med rektanglar (objekt av typen Rectangle). Använd sedan metoderna selectRectangles och minRectangle för att bestämma den minsta av de rektanglar i vektorn som är gula.

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

Klassen CharSequence hanterar en teckensekvens:

class CharSequence

{

private static final int INITIAL\_CAPACITY = 3;

// tecken i teckensekvensen

private char[] chars;

// antalet tecken

private int charCount;

public CharSequence ()

{

this.chars = new char[INITIAL\_CAPACITY];

this.charCount = 0;

}

public String toString ()

{

return "[" + chars.length + ", " + new String (chars, 0, charCount) + "]";

}

// add lägger till ett givet tecken i teckensekvensen.

// koden saknas här

// replace ersätter det tecken som finns på ett givet index

// med ett givet tecken.

// I fall att ett ogiltigt index anges, kastas ett undantag

// av typen java.lang.IndexOutOfBoundsException.

// koden saknas här

}

En instans av klassen CharSequence skapas och används så här:

CharSequence cs = new CharSequence ();

for (char c = 'a'; c <= 'e'; c++)

{

cs.add (c);

System.out.println (cs);

}

cs.replace (1, 'B');

System.out.println (cs);

cs.replace (5, 'B');

När detta kodavsnitt exekveras, skapas följande utskrift:

[3, a]

[3, ab]

[3, abc]

[6, abcd]

[6, abcde]

[6, aBcde]

Exception in thread "main" java.lang.IndexOutOfBoundsException: bad index

at CharSequence.replace(UseCharSequence.java:49)

at UseCharSequence.main(UseCharSequence.java:69)

a) Implementera metoden add.

b) Implementera metoden replace.

c) Hur ser ut det objekt som refereras med referensen cs, när kodavsnittet har exekverats? Rita objektet.

Uppgift 4 (4 poäng + 5 poäng)

Klassen MessageQueue definierar en kö för meddelanden. Klasserna LifoMessageQueue och FifoMessageQueue representerar två olika typer av köer.

abstract class MessageQueue

{

protected String[] messages;

public MessageQueue ()

{

messages = new String[5];

}

// put sätter in ett givet meddelande i kön.

public abstract void put (String message);

// take tar ut nästa meddelande ur kön och returnerar det.

public abstract String take ();

public String queueStructure ()

{

String s = "";

for (int index = 0; index < messages.length; index++)

if (messages[index] == null)

s = s + "-";

else

s = s + messages[index];

return s;

}

}

class LifoMessageQueue extends MessageQueue

{

private int lastIndex = -1;

public LifoMessageQueue ()

{

super ();

}

public String toString ()

{

String s = "";

for (int index = 0; index <= lastIndex; index++)

s = s + messages[index];

return s;

}

public void put (String message)

{

lastIndex = lastIndex + 1;

messages[lastIndex] = message;

}

public String take ()

{

String lastMessage = messages[lastIndex];

messages[lastIndex] = null;

lastIndex--;

return lastMessage;

}

}

class FifoMessageQueue extends MessageQueue

{

private int numberOfMessages = 0;

private int firstIndex = 0;

private int lastIndex = -1;

public FifoMessageQueue ()

{

super ();

}

public String toString ()

{

String s = "";

int index = firstIndex;

for (int i = 1; i <= numberOfMessages; i++)

{

s = s + messages[index];

index = (index + 1) % messages.length;

}

return s;

}

public void put (String message)

{

lastIndex = (lastIndex + 1) % messages.length;

messages[lastIndex] = message;

numberOfMessages++;

}

public String take ()

{

String firstMessage = messages[firstIndex];

messages[firstIndex] = null;

firstIndex = (firstIndex + 1) % messages.length;

numberOfMessages--;

return firstMessage;

}

}

a)

MessageQueue mq = new LifoMessageQueue ();

mq.put ("A");

mq.put ("B");

mq.put ("C");

mq.put ("D");

String m = mq.take ();

m = mq.take ();

mq.put ("E");

System.out.println (mq);

System.out.println (mq.queueStructure ());

Vilken utskrift skapas när det här kodavsnittet utförs?

b)

MessageQueue mq = new FifoMessageQueue ();

mq.put ("a");

mq.put ("b");

mq.put ("c");

String m = mq.take ();

mq.put ("d");

mq.put ("e");

mq.put ("f");

m = mq.take ();

m = mq.take ();

System.out.println (mq);

System.out.println (mq.queueStructure ());

Vilken utskrift skapas när det här kodavsnittet utförs?

Uppgift 5 (5 poäng + 4 poäng)

En algoritm, som sorterar en sekvens med element, kan illustreras som nedan:

[H, B, G, A, E, F, C, D]

[B, H, G, A, E, F, C, D]

[B, G, H, A, E, F, C, D]

[A, B, G, H, E, F, C, D]

[A, B, E, G, H, F, C, D]

[A, B, E, F, G, H, C, D]

[A, B, C, E, F, G, H, D]

[A, B, C, D, E, F, G, H]

[A, B, C, D, E, F, G, H]

a) Skapa en metod sort som tar emot en vektor med teckensträngar, och sorterar den enligt givna algoritmen.

b) Låt *n* beteckna antalet element som sorteras. Bestäm i så fall tidskomplexiteten för algoritmen när det gäller antalet jämförelser - både i bästa fall och i värsta fall. Kategorisera motsvarande komplexitetsfunktioner: till vilken *Θ* –mängd tillhör dem?