

Tentamen – obligatorisk del

Förklaringar

Den här delen av tentamen är obligatorisk och tillräcklig för betyget E

För att klara tentamen, måste studenten klara den här delen av tentamen. Om bara den här delen klaras, blir betyget på tentamen E. För att uppnå ett högre betyg, måste studenten samla ett tillräckligt antal poäng även på den extra delen av tentamen. Beroende på detta extra antal poäng, kan betyget på tentamen bli D, C, B eller A.

För att klara den här delen av tentamen

För att klara denna obligatoriska del av tentamen, måste studenten uppnå minst två tredjedelar av det totala antalet poäng på den här delen. Dessa poäng bidrar inte till de högre betygen. Det högsta betyget på denna obligatoriska del är E.

Antalet poäng

Totalt: 33 poäng

För betyget E krävs minst: 22 poäng

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 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.

Uppgifter

Uppgift 1 (2 poäng + 2 poäng)

```
int[]    u = {1, 2, 3, 4, 5};
int      e = u[0];
for (int pos = 0; pos < u.length - 1; pos++)
    u[pos] = u[pos + 1];
u[u.length - 1] = e;

int[]    k = {1, 2, 3, 4};
Integer[] v = new Integer[4];
for (int pos = 0; pos < v.length; pos++)
    v[pos] = new Integer (k[pos] % 2);
```

- a) Rita den vektor som refereras med referensen *u*.
- b) Rita den vektor som refereras med referensen *v*.

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

Om mängden av reella tal betecknas med *R* och mängden av hela tal med *Z*, kan matematiska funktioner *signum*, *abs* och *floor* definieras så här:

signum: $R \rightarrow Z$
 $x < 0$: *signum* (*x*) = -1

$x = 0$: $\text{signum}(x) = 0$
 $x > 0$: $\text{signum}(x) = 1$

$\text{abs}: \mathbb{R} \rightarrow \mathbb{R}$
 $x < 0$: $\text{abs}(x) = -x$
 $x \geq 0$: $\text{abs}(x) = x$

$\text{floor}: \mathbb{R} \rightarrow \mathbb{Z}$
 $\text{floor}(x) = \max \{ m \in \mathbb{Z} \mid m \leq x \}$

Skapa en klass `Math`, som innehåller tre statiska metoder `signum`, `abs` och `floor`. Dessa metoder implementerar motsvarande matematiska funktioner.

Uppgift 3 (3 poäng + 4 poäng)

En klass `Word` representerar ett ord, både på svenska och på engelska:

```
public class Word
{
    // ordet på svenska
    private String sWord;

    // ordet på engelska
    private String eWord;

    public Word (String sWord, String eWord)
    {
        this.sWord = sWord;
        this.eWord = eWord;
    }

    public String wordSw ()
    {
        return this.sWord;
    }

    public String wordEn ()
    {
        return this.eWord;
    }

    public String toString ()
    {
        return this.sWord + " | " + this.eWord;
    }
}
```

En statisk metod, `merge`, tar emot två vektorer med ord. Metoden returnerar en annan vektor, som innehåller alla givna ord.

```
public static Word[] merge (Word[] words1, Word[] words2)
{
    // koden saknas här
}
```

a) Skapa metoden `merge`.

b) Skapa två vektorer med ord, och anropa metoden `merge` i samband med dem. Visa sedan den vektor som returneras. Utskriften ska vara på följande form:

```
ett | one
två | two
tre | three
fyra | four
fem | five
sex | six
sju | seven
åtta | eight
```

Uppgift 4 (2 poäng + 3 poäng + 2 poäng + 1 poäng)

Klassen `StringCreator` representerar en föränderlig teckensträng:

```
public class StringCreator
{
    // tecken i teckensträngen
    private char[] chars;
    // antalet tecken
    private int charCount;

    // StringCreator skapar en teckensträng utifrån
    // en given teckenvektor.
    public StringCreator (char[] chars)
    {
        this.chars = new char[chars.length + 1];

        // kopiera tecken
        // koden saknas här

        // justera variabeln charCount
        // koden saknas här
    }

    // insert sätter in ett givet tecken på en given position
    // i den här teckensträngen
    public void insert (char c, int pos)
        throws ArrayIndexOutOfBoundsException
    {
        if (pos > charCount)
            throw new ArrayIndexOutOfBoundsException ("bad index: " + pos);

        // koden saknas här
        // utöka inte teckenvektorn
    }

    public String toString ()
    {
        return new String (chars, 0, charCount);
    }
}
```

En teckensträng skapas och används så här:

```
char[] digits = {'1', '2', '3', '4'};
StringCreator sc = new StringCreator (digits);
int pos = 0;
// pos = 5; // (1)
sc.insert ('+', pos);
System.out.println (sc);
```

När detta kodavsnitt exekveras, skapas följande utskrift:

```
+1234
```

- a) Implementera konstruktorn `StringCreator`.
- b) Implementera metoden `insert`.
- c) Rita det objekt som refereras med referensen `sc`.
- d) Vad händer om satsen (1) inkluderas i det givna kodavsnittet?

Uppgift 5 (4 poäng + 2 poäng + 2 poäng)

Klassen `Shape` representerar en geometrisk figur:

```
public abstract class Shape
```

```

{
    // figurens färg
    private String    colour;

    public Shape (String colour)
    {
        this.colour = colour;
    }

    public String getColour ()
    {
        return colour;
    }

    // area returnerar figurens area
    public abstract double area ();
}

```

Klasserna `Rectangle` och `Circle` representerar två subklasser till klassen `Shape`. Klassen `Circle` är implementerad så här:

```

class Circle extends Shape
{
    // cirkelns radie
    private double    radius;

    // Circle skapar en cirkel - cirkelns färg och radie är givna
    public Circle (String colour, double radius)
    {
        super (colour);
        this.radius = radius;
    }

    // koden saknas här
}

```

Flera figurer skapas och används så här:

```

Shape[]    shapes = new Shape[4];
shapes[0] = new Circle ("blue", 10);
shapes[1] = new Rectangle ("yellow", 3, 4);
shapes[2] = new Rectangle ("yellow", 4, 5);
shapes[3] = new Circle ("blue", 5);
// shapes[0] = new java.lang.String ("red"); // (1)

for (Shape shape : shapes)
{
    double    ar = shape.area ();
    System.out.println (shape + " : " + ar);
}

```

När detta kodavsnitt exekveras, skapas följande utskrift:

```

[blue | (10.0)] : 314.1592653589793
[yellow | (3.0, 4.0)] : 12.0
[yellow | (4.0, 5.0)] : 20.0
[blue | (5.0)] : 78.53981633974483

```

- Komplettera klassen `Circle`: skriv den kod som saknas.
- Vad händer när satsen (1) inkluderas? Varför?
- Rita det objekt som refereras med referensen `shapes[1]`.