LUNDS TEKNISKA HÖGSKOLA Institutionen för datavetenskap

Kontrollskrivning 3, Programmeringsteknik för D/C

2011–12–12, 8.00–13.00

Anvisningar: Preliminärt ger uppgifterna $8 + 10 + 5 + 17 = 40$ poäng. Tillåtet hjälpmedel: Java-snabbreferens.

Jag meddelar på kurshemsidan (`cs.lth.se/EDA016`) när rättningen är klar (före jul). Då kan du kontakta mig via e-post (`Per.Holm@cs.lth.se`) för att få reda på ditt resultat.

Omkontrollskrivning på hela kursen äger rum tisdag 10 januari 2012, 8.00–13.00 i Sparta B. Det behövs inte någon föranmälan till omkontrollskrivningarna. Det är tillåtet att försöka höja sitt betyg (”plussa”) vid en omkontrollskrivning.

---

1. Om man har tillgång till en slumptalsgenerator som ger rektangelfördelade slumptal[1] kan man med följande algoritm[2] generera *två* normalfördelade slumptal z_0 och z_1 med medelvärdet 0 och standardavvikelsen 1:

   1. Dra två rektangelfördelade slumptal i intervallet $(0, 1]$ (0 ingår inte i intervallet, 1 ingår). Kalla talen u_0 och u_1.
   2. Beräkna $r = \sqrt{-2 \ln u_0}$ och $\theta = 2\pi u_1$.
   3. $z_0 = r \cos\theta$, $z_1 = r \sin\theta$.

   Implementera en klass `NormalRandom` med en metod `nextNormal()` som vid varje anrop ger ett normalfördelat slumptal, beräknat med ovanstående algoritm. Observera att algoritmen bara ska köras vid vartannat anrop av metoden.

[1] Slumptal är rektangelfördelade om alla tal är lika sannolika.

[2] Se `http://en.wikipedia.org/wiki/Box-Muller_transform`.

2. I vissa bloggar kan författaren förse sina inlägg med ”taggar” som talar om vad inlägget handlar om. En tagg är *ett* ord. Implementera följande klass, som beskriver ett blogginlägg:

   `BlogPost`

```
/** Skapar ett blogginlägg med titeln title, texten text och publiceringsdatum
    date (på formen yyyy-mm-dd) */
BlogPost(String title, String text, String date);

/** Lägger till taggen tag, om den inte redan finns */
void addTag(String tag);

/** Undersöker om alla orden i listan words ingår bland inläggets taggar.
    Returnerar true i så fall, annars false */
boolean allTagsMatch(ArrayList<String> words);

/** Returnerar en sträng med titel, datum och text. Mellan titel och datum
    ska det vara ett blanktecken, mellan datum och text ny rad ("\n") */
String toString();
```

3. Alla blogginläggen samlas i ett objekt av klassen `Blog`:

```
public class Blog {
    private ArrayList<BlogPost> posts;

    /** Skapar en blogg */
    public Blog() {
        posts = new ArrayList<BlogPost>();
        // ... här läses blogginläggen från en databas och lagras i
        // ... listan posts
    }

    /** Ger en lista med de blogginlägg för vilka det gäller att alla orden
        i listan words ingår bland inläggets taggar */
    public ArrayList<BlogPost> getMatchingPosts(ArrayList<String> words) {
        // ... implementera
    }
}
```

   Implementera operationen `getMatchingPosts`.

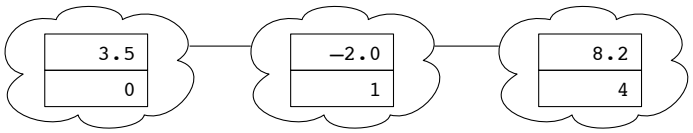
4. En klass som beskriver polynom i en variabel har följande specifikation (bara några få av operationerna har tagits med):

```
/** Skapar ett tomt polynom */
Polynom();

/** Lägger in en ny term med koefficienten coeff och gradtalet degree
    i polynomet. Om det redan finns en term med detta gradtal ska det
    nya koefficientvärdet adderas till det gamla */
void addTerm(double coeff, int degree);

/** Beräknar polynomets värde i punkten x */
double getValue(double x);

/** Returnerar derivatan av polynomet */
Polynom differentiate();

/** Skriver ut polynomet med en term (koefficient och gradtal) per rad */
void print();
```

Exempel: `getValue(0)` på polynomet $3.5 - 2x + 8.2x^4$ ger resultatet 3.5, `getValue(2)` ger resultatet 130.7 ($3.5 - 2 \cdot 2 + 8.2 \cdot 2^4$). Operationen `differentiate` på detta polynom ger det nya polynomet $-2 + 4 \cdot 8.2 \cdot x^3$.

I klassen ska polynomet representeras genom att koefficient och gradtal för de termer vars koefficient inte är noll lagras i en lista, som hela tiden *ska* vara sorterad efter växande gradtal.

Till exempel representeras polynomet $3.5 - 2x + 8.2x^4$ av följande lista:



Elementen i listan är objekt av den färdigskrivna klassen `Term`:

```
/** Skapar en term med koefficienten coeff och gradtalet degree */
Term(double coeff, int degree);

/** Tar reda på koefficientvärdet */
double getCoeff();

/** Adderar x till koefficientvärdet */
void addToCoeff(double x);

/** Tar reda på gradtalet */
int getDegree();
```

Implementera klassen `Polynom`. Tänk på att det aldrig får finnas termer med koefficienten noll i listan.