Kontrollskrivning 3, Programmeringsteknik för D/C

2013–12–16, 8.00–13.00

Anvisningar: Preliminärt ger uppgifterna $11 + 11 + 18 = 40$ poäng. Tillåtet hjälpmedel: Java-snabbreferens.

Jag meddelar på kurshemsidan, `http://cs.lth.se/eda016`, när rättningen är klar (jag hoppas vara 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 fredag 17 januari 2014 kl 8–13 i Sparta A. Det behövs inte någon föranmälan till omkontrollskrivningarna. Det är tillåtet att försöka höja sitt betyg (”plussa”) vid en omkontrollskrivning.

---

1. Skriv en klass `Urand` som genererar slumpmässiga heltal i ett intervall $[0, n)$. Slumptalen ska dras *utan* återläggning, det vill säga man ska få varje tal i intervallet bara en gång. När man har fått alla tal i intervallet ska man börja om från början.

   Klassen har följande specifikation:

   ```
   /** Skapar en slumptalsgenerator som genererar tal i intervallet [0,n)
       utan återläggning */
   URand(int n);

   /** Returnerar nästa slumptal */
   int nextInt();
   ```

   Exempel på användning av klassen:

   ```
   public static void main(String[] args) {
       URand urand = new URand(10);
       for (int i = 0; i < 5; i++) {
           for (int j = 0; j < 10; j++) {
               System.out.print(urand.nextInt() + " ");
           }
           System.out.println();
       }
   }
   ```

   Programmet genererar fem rader utskrift; varje rad innehåller talen 0–9 i slumpmässig ordning. Råd: lägg in talen $0, 1, \ldots, n-1$ i en vektor och hämta tal ur vektorn efterhand. Blanda talen i vektorn när det behövs. Använd följande metod för att blanda:

   ```
   för i = n-1, n-2, ..., 2, 1 {
       j = slumptal i intervallet [0,i+1)
       byt plats på talen med index i och index j
   }
   ```

2. I ett program läser man kommandon och heltal och summerar talen. Exempel på konversation med programmet (kommentarerna ingår inte; `p`, `a`, `u`, `c` och `r` är kommandon):

```
p       // print the sum, 0 to begin with
The sum is 0
a 4     // add 4
a 3     // add 3
a 2     // add 2
p
The sum is 9
u       // undo last add
u       // multiple undo's are allowed
p
The sum is 4
c       // commit changes (make them permanent so they cannot be undone)
u       // nothing happens
a 3
a 2
p
The sum is 9
r       // rollback, undo all changes since last commit
p
The sum is 4
```

Programmet ser ut så här:

```
public static void main(String[] args) {
    Accumulator accum = new Accumulator();
    Scanner scan = new Scanner(System.in);
    while (scan.hasNext()) {
        char command = scan.next().charAt(0);
        switch (command) {
        case 'p': System.out.println("The sum is " + accum.getSum()); break;
        case 'a': int nbr = scan.nextInt(); accum.add(nbr); break;
        case 'u': accum.undo(); break;
        case 'c': accum.commit(); break;
        case 'r': accum.rollback(); break;
        }
    }
}
```

Implementera klassen `Accumulator`. Ledning: för att man ska kunna göra ”undo” och ”rollback” behöver man en lista där man sparar talen som adderas.

3. I många program behöver man hålla reda på olika ”egenskaper”. För ett program med ett grafiskt användargränssnitt kan det till exempel gälla fönstrets position på skärmen och fönstrets bakgrundsfärg. Vi förutsätter att alla egenskaper är strängar och att de har namn som också är strängar.

   För varje program finns det ett antal ”standardegenskaper” som gäller för alla användare av programmet. De egenskaperna beskrivs i en fil. Exempel:

   ```
   windowX=100
   windowY=250
   windowcolor=white
   linewidth=1
   ```

   En användare kan vilja ändra på egenskaperna eller lägga till nya egenskaper, men de ändringarna får inte påverka standardegenskaperna. Därför sparas sådana ändringar i en fil som användaren äger. Exempel (två egenskaper har ändrats, en egenskap har lagts till):

   ```
   linewidth=3
   windowcolor=lightgray
   linecolor=red
   ```

   När man ska hantera egenskaperna i ett program gör man det i en klass `Properties` (klassen som beskrivs här är en förenklad version av standardklassen `java.util.Properties`). Klassen har följande specifikation:

   ```
   /** Skapar ett tomt Properties-objekt. Parametern defaults innehåller
       standardegenskaperna */
   Properties(Properties defaults);

   /** Skapar ett tomt Properties-objekt utan något defaults-objekt */
   Properties();

   /** Tar reda på egenskapen med namnet key. Om key inte finns returneras det
       värde som finns i defaults-objektet. Om key inte finns där heller
       returneras null */
   String getProperty(String key);

   /** Sparar egenskapen value med namnet key. Om key redan finns skrivs det
       gamla värdet över */
   void setProperty(String key, String value);

   /** Skriver ut de lagrade namnen och egenskaperna på strömmen file, med
       ett namn och en egenskap per rad, åtskilda med likhetstecken */
   void store(PrintWriter file);

   /** Läser in namn och egenskaper med hjälp av scannern scan. Scannern är
       kopplad till en fil som har skrivits av metoden store */
   void load(Scanner scan);
   ```

   I `main`-metoden på nästa sida visas hur man kan använda klassen. Standardegenskaperna finns i filen `defaults.txt`, de användarspecifika egenskaperna i filen `prop.txt`. Först skapas två `Properties`-objekt, ett för standardegenskaperna och ett för de användarspecifika egenskaperna. Sedan skapas nya användarspecifika egenskaper, och till sist testas att sökningen efter egenskaper fungerar.

. . . forts nästa sida

```
public class PropertiesTest {
    public static void main(String[] args) throws FileNotFoundException {
        Properties defaults = new Properties();
        defaults.load(new Scanner(new File("defaults.txt")));
        Properties prop = new Properties(defaults);
        prop.load(new Scanner(new File("prop.txt")));

        Scanner scan = new Scanner(System.in);
        System.out.println("Enter key and value to insert, q to finish");
        String key = scan.next();
        while (!key.equals("q")) {
            String value = scan.next();
            prop.setProperty(key, value);
            key = scan.next();
        }

        System.out.println("Enter key to search for, q to finish");
        key = scan.next();
        while (!key.equals("q")) {
            System.out.println(key + " = " + prop.getProperty(key));
            key = scan.next();
        }
        prop.store(new PrintWriter(new File("prop.txt")));
    }
}
```

Implementera klassen. Du kan förutsätta att alla parametrar är korrekta. Råd: skriv en klass `Entry` som innehåller ett `key`-värde och ett `value`-värde. Lagra `Entry`-objekt i en lista i `Properties`-klassen.

Antag att standardegenskaperna och de användarspecifika egenskaperna är som i det inledande exemplet. Följande tabell visar resultatet av några sökningar:

Anrop	Resultat	Kommentar
`prop.getProperty("windowcolor")`	`"lightgray"`	hittas i `prop`
`prop.getProperty("linecolor")`	`"red"`	hittas i `prop`
`prop.getProperty("windowX")`	`"100"`	finns inte i `prop`, hittas i `defaults`
`prop.getProperty("linetype")`	`null`	finns inte i `prop` och inte i `defaults`