

EDA016 Programmeringsteknik för D

Läsvecka 5: Tecken, strängar och slumpal

Björn Regnell

Datavetenskap, LTH

Lp1-2, HT 2015

5 Tecken, strängar och slumpstal

- Att göra denna vecka
- Tecken
- Strängar
- Slumptal
- do-while
- Formatering och utskrift
- Switch

Att göra i Vecka 5: Förstå aritmetiska och logiska uttryck, använda klasser mha klass-specifikationer

- 1 Läs följande kapitel i kursboken: 11, 7.9, 6.10, 7.7, 7.4, 7.12
Begrepp: sträng, toString, StringBuilder, slumpal, Random
- 2 Gör övning 5: Klasser, slumpal
- 3 Gör gammal kontrollskrivning & rätta i samarbetsgrupper
- 4 Gör Lab 4: implementera square

Hur använda föreläsningsexempel

- Föreläsningsexempel för användning i Eclipse finns här:
<https://github.com/bjornregnell/lth-eda016-2015/tree/master/lectures/examples/eclipse-ws>
- Följ instruktionerna i **README.md** för att öppna exemplen i ett eget workspace
- Ändra, prova varianter, gör nya testprogram, etc. Det är genom att **aktivt koda** som du lär dig!

Tecken

Specialtecken

Some **Java Escape Sequences**:

- Ny rad: `\n`
- Tab: `\t`
- Citationstecken: `\'`
- Tab: `\"`
- Godtyckligt unicode-tecken: `\u03BB`

Tecken

```
1 package week05;
2
3 public class ShowCharacters {
4
5     public static void showChar(char c){
6         System.out.print("'" + c + "'" + " == " + (int) c + " ");
7     }
8
9
10    public static void main(String[] args) {
11
12        for (char c = '0'; c <= '9'; c++) {
13            showChar(c);
14        }
15
16        System.out.print("\n\n");
17
18        for (char c = 32; c <= 1000; c++) {
19            showChar(c);
20            if ((c - ' ' + 1) % 10 == 0) {
21                System.out.print("\n");
22            }
23        }
24
25        System.out.println("\n\n unicode 955: \u03BB");
26    }
27 }
```

Klassen Character

Klassen **Character** innehåller många användbara metoder, t.ex.:

Character

```
/** Determines if the specified character is a digit. */  
static boolean isDigit(char ch);  
  
/** Determines if the specified character is a letter. */  
static boolean isLetter(char ch);  
  
/** Determines if the specified character is white space. */  
static boolean isWhitespace(char ch);  
  
/** Determines if the specified character is an uppercase character. */  
static boolean isUpperCase(char ch);  
  
/** Determines if the specified character is a lowercase character. */  
static boolean isLowerCase(char ch);  
  
/** Converts the character argument to uppercase */  
static char toUpperCase(char ch);  
  
/** Converts the character argument to lowercase */  
static char toLowerCase(char ch);
```


Strängar

String och StringBuilder

Standardklasser (i paketet `java.lang`, behöver inte importeras):

String Beskriver en följd av tecken. Tecknen kan läsas av men inte ändras. Med operatoren `+` konkatenerar man (slår ihop) två strängar, eller en sträng med ett talvärde. Då bildas ett nytt strängobjekt.

StringBuilder En följd av tecken som kan läsas av *och* ändras.

```
String s1 = "En text";  
String s2 = "en text till";  
String result = s1 + " och " + s2;  
  
int x = 10;  
int y = 30;  
String s1 = "Summan är " + x + y;  
String s2 = "Summan är " + (x + y);
```

Viktiga operationer på String

String

```
String();           // skapar en tom sträng
                   // (kan också skrivas "")
int length();       // antalet tecken
char charAt(int pos); // tecknet på plats pos
boolean equals(Object s); // true om innehållet i
                   // aktuell sträng är lika
                   // med innehållet i s
int compareTo(String s); // jämför aktuell sträng med s
int indexOf(char ch); // index för den första
                   // förekomsten av ch, -1
                   // om ch inte finns
String substring(int start, // ny sträng med tecknen
                 int end); // med index [start, end)
```

```
"Java".compareTo("Java") == 0
"java".compareTo("javac") < 0
"java".compareTo("Java") > 0
"java".compareTo("jazz") < 0
```

Användning av String, 1

```
public class Text {  
    private String s;  
  
    public Text(String s) {  
        this.s = s;  
    }  
  
    /** Tar reda på antalet blanktecken i strängen */  
    public int getNbrSpaces() {  
        int spaces = 0;  
        for (int i = 0; i < s.length(); i++) {  
            if (s.charAt(i) == ' ') {  
                spaces++;  
            }  
        }  
        return spaces;  
    }  
}
```

Användning av String, 2

```
/** Tar reda på index för den första förekomsten av tecknet  
    ch i texten, -1 om inget sådant tecken finns */  
public int indexOf(char ch) {  
    int i = 0;  
    while (i < s.length() && s.charAt(i) != ch) {  
        i++;  
    }  
    return (i < s.length()) ? i : -1;  
}
```

Användning av String, 3

```
/** Tar reda på det första ordet i texten. Ett ord är en
    följd av tecken som inte är blanka */
public String firstWord() {
    int start = 0;
    while (start < s.length() &&
           Character.isWhitespace(s.charAt(start))) {
        start++;
    }
    int end = start;
    while (end < s.length() &&
           ! Character.isWhitespace(s.charAt(end))) {
        end++;
    }
    return s.substring(start, end);
}
```

StringBuilder

Skapa, ta reda på längd och tecken (som String):

StringBuilder

```
StringBuilder();           // tom strängbuffert
StringBuilder(String s);   // kopia av s
int length();              // antalet tecken
char charAt(int pos);      // tecknet på plats pos
String toString();         // skapar ett String-objekt med
                           // samma innehåll som denna
                           // strängbuffert
```

Ändra innehållet i StringBuilder-objekt

StringBuilder

```
void setCharAt(int k, char ch); // ändrar tecknet på
                                // plats k till ch
StringBuilder append(String s); // lägger till s
                                // efter tecknen
StringBuilder insert(int k, String s); // lägger in s
                                // på plats k, flyttar
                                // tecknen efter
StringBuilder deleteCharAt(int k); // tar bort tecknet
                                // på plats k
StringBuilder delete(int start, int end); // tar bort
                                // tecknen [start,end)
StringBuilder replace(int start, int end, String s);
                                // ersätter tecknen
                                // [start,end) med s
```


Användning av StringBuilder, 1

```
public class MutableText {  
    private StringBuilder sb;  
  
    public MutableText(String s) {  
        sb = new StringBuilder(s);  
    }  
  
    /** Ändrar alla små bokstäver a-z i texten till motsvarande stora */  
    public void changeToUpperCase() {  
        for (int i = 0; i < sb.length(); i++) {  
            char ch = sb.charAt(i);  
            if (ch >= 'a' && ch <= 'z') {  
                sb.setCharAt(i, (char) (ch - 'a' + 'A'));  
            }  
        }  
    }  
}
```

Användning av StringBuilder, 2

```
/** Lägger in ett blanktecken efter varje punkt och kommatecken  
    i texten, dock ej efter textens sista tecken */  
public void insertSpaces() {  
    int i = 0;  
    while (i < sb.length() - 1) {  
        if (sb.charAt(i) == '.' || sb.charAt(i) == ',') {  
            sb.insert(i + 1, ' ');  
            i++;  
        }  
        i++;  
    }  
}
```

Ta bort blanktecken från sträng

```
public static String removeSpacesSlow(String s) { // "DÅLIGT"
    String result = "";
    for (int i = 0; i < s.length(); i++) {
        if (s.charAt(i) != ' ') {
            result += s.charAt(i);
        }
    }
    return result;
}

public static String removeSpaces(String s) { // "BRA"
    StringBuilder sb = new StringBuilder();
    for (int i = 0; i < s.length(); i++) {
        if (s.charAt(i) != ' ') {
            sb.append(s.charAt(i));
        }
    }
    return sb.toString();
}
```

Exempel: Nananananana Batman!

<https://www.youtube.com/watch?v=oDc-1zfffMw>

Prova **detta exempel** på din dator och se hur snabbt det går för stora n med String versus StringBuilder:

```
*** MEASURING n == 16
Singing Batman with String:      Timer measured: 0 ms
Singing Batman with StringBuilder: Timer measured: 0 ms
```

Hur snabb är din dator? Så här snabb är min i7-4790K

```
*** MEASURING n == 1024
  Singing Batman with String:      Timed: less than 7 ms
  Singing Batman with StringBuilder: Timed: less than 1 ms
*** MEASURING n == 2048
  Singing Batman with String:      Timed: less than 24 ms
  Singing Batman with StringBuilder: Timed: less than 1 ms
*** MEASURING n == 4096
  Singing Batman with String:      Timed: less than 74 ms
  Singing Batman with StringBuilder: Timed: less than 1 ms
*** MEASURING n == 8192
  Singing Batman with String:      Timed: less than 241 ms
  Singing Batman with StringBuilder: Timed: less than 1 ms
*** MEASURING n == 16384
  Singing Batman with String:      Timed: less than 873 ms
  Singing Batman with StringBuilder: Timed: less than 1 ms
*** MEASURING n == 32768
  Singing Batman with String:      Timed: less than 4269 ms
  Singing Batman with StringBuilder: Timed: less than 1 ms
*** MEASURING n == 65536
  Singing Batman with String:      Timed: less than 17877 ms
  Singing Batman with StringBuilder: Timed: less than 1 ms
*** MEASURING n == 131072
  Singing Batman with String:      Timed: less than 78529 ms
  Singing Batman with StringBuilder: Timed: less than 2 ms
*** MEASURING n == 262144
  Singing Batman with String:      Timed: less than 376264 ms
  Singing Batman with StringBuilder: Timed: less than 4 ms
```

toString

```
public class Complex {  
    private double re; // realdel  
    private double im; // imaginärdel  
  
    public Complex(double re, double im) {  
        this.re = re;  
        this.im = im;  
    }  
  
    public String toString() { //vad händer i exemplet om du tar bort denna?  
        return "Complex(" + re + ", " + im + ")";  
    }  
}
```

Exempel på explicit och **implicit** användning av toString:

```
Complex z = new Complex(1.5, 2.3);  
System.out.println("z = " + z.toString());  
System.out.println("z = " + z);
```

Programexempel: Datakomprimering

Användning av s.k. följdängdskodning

Krav

- 1 Man räknar hur många gånger som ett tecken förekommer i följd.
- 2 Om antalet är större än 3 lagras först ett dollartecken, sedan antalet tecken, sedan tecknet. Dollartecknet fungerar som ett "escapetecken" som talar om att de följande tecknen ska tolkas på ett speciellt sätt.
- 3 Om antalet är mindre än eller lika med 3 lagras alla tecken.
- 4 Exempel: *aabbbbcbdddeeeeeffff* kodas som *aa\$5bcddd\$6e\$4f*. Siffrorna är inte tecknen '5', '6' och '4' utan tecknen med Unicode-numren 5, 6 respektive 4.
- 5 Förenklingar:
 - Vi förutsätter att det inte finns några dollartecken i texten.
 - Vi förutsätter att det inte finns fler tecken i rad än att antalet ryms i en **char**-variabel (16 bitar).

Compressor specifikation och design

Compressor

```
/** komprimerar klartextsträngen s med följdängdskodning */  
public static String compress(String s)  
  
/** dekomprimerar följdängdskodade strängen s till klartext */  
public static String decompress(String s)
```

Design av compress:

pseudokod:

- så länge strängen inte är slut:
 - så länge alla tecken lika:
räkna antalet lika tecken i följd
 - om fler än 3 lika:
bygg på sträng med komprimerad följd
annars: bygg på sträng med okomprimerad följd

Komprimering

```
public static String compress(String s) {  
    StringBuilder sb = new StringBuilder();  
    int i = 0;  
    while (i < s.length()) {  
        char ch = s.charAt(i);  
        int nbrEqual = 1;  
        i++;  
        while (i < s.length() && s.charAt(i) == ch) {  
            i++;  
            nbrEqual++;  
        }  
        if (nbrEqual > 3) {  
            sb.append('$');  
            sb.append((char) nbrEqual);  
            sb.append(ch);  
        } else {  
            for (int k = 0; k < nbrEqual; k++) {  
                sb.append(ch);  
            }  
        }  
    }  
    return sb.toString();  
}
```

Dekomprimering

```
public static String decompress(String s) {  
    StringBuilder sb = new StringBuilder();  
    int i = 0;  
    while (i < s.length()) {  
        char ch = s.charAt(i);  
        if (ch != '$') {  
            sb.append(ch);  
        } else {  
            i++;  
            int nbrEqual = s.charAt(i);  
            i++;  
            ch = s.charAt(i);  
            for (int k = 0; k < nbrEqual; k++) {  
                sb.append(ch);  
            }  
            i++;  
        }  
    }  
    return sb.toString();  
}
```

Slumptal

Slumptal

- **Slumptalsgenerering** är ett viktigt område inom mjukvaruutveckling, speciellt inom kryptering, intrångsskydd, simulering och spelutveckling.
- Slumptal får man i Java med hjälp av standardklassen `java.util.Random`

Random

```
/** En slumptalsgenerator med slumptalsfröet seed */  
Random(long seed);  
  
/** En slumptalsgenerator med ett slumpmässigt  
    slumptalsfrö */  
Random();  
  
/** Slumpmässigt heltal i intervallet [0,n) */  
int nextInt(int n);  
  
/** Slumpmässigt reellt tal i intervallet [0,1.0) */  
double nextDouble();
```

Användning av Random

10 slumpmässiga heltal i intervallet [1, 6], 10 reella tal i intervallet [5.0, ~15.0):

```
import java.util.Random;

public class RandomExample {
    public static void main(String[] args) {
        Random rand = new Random();
        for (int i = 0; i < 10; i++) {
            int iRand = 1 + rand.nextInt(6);
            System.out.println(iRand);
        }
        for (int i = 0; i < 10; i++) {
            double dRand = 5 + 10 * rand.nextDouble();
            System.out.println(dRand);
        }
    }
}
```

Programexempel: Tärningsspel

Krav

- 1 Ett tärningsspel med två spelare ska simuleras i ett program.
- 2 Den förste spelaren kastar tärningen och räknar antalet kast tills det i två kast i följd blir samma antal prickar på tärningen.
- 3 Därefter kastar den andre spelaren tärningen på samma sätt.
- 4 Den av spelarna som gjort minst antal kast vinner spelet.
- 5 Om båda spelarna gjort samma antal kast kastar de båda igen tills någon av dem vunnit.
- 6 När spelet är klart ska namnet på vinnaren skrivas ut.

Design: Dela upp koden i dessa klasser:

- Die har hand om data och operationer för en **tärning**
- Player har hand om data och operationer för en **spelare**
- DiceGame **genomför ett spel**

main-metod som genomför spelet

Test: Ett huvudprogram som genomför spelet:

```
public class PlayGame {  
    public static void main(String[] args) {  
        Player p1 = new Player("Kim");  
        Player p2 = new Player("Robin");  
        DiceGame game = new DiceGame(p1, p2);  
        Player winner = game.play();  
        System.out.println(winner.getName() + " vann");  
    }  
}
```

Testresultat: Utskrift av vinnarens namn.

Klassen Die, en tärning

```
import java.util.Random;
public class Die {
    private static Random rand = new Random();
    private int pips;

    /** Skapar en tärning */
    public Die() {
        roll(); // så att pips får ett värde 1..6
    }

    /** Kastar tärningen */
    public void roll() {
        pips = 1 + rand.nextInt(6);
    }

    /** Tar reda på resultatet av det senaste kastet */
    public int getResult() {
        return pips;
    }
}
```


Specifikation av Player, en spelare

Båda spelarna ska spela med samma tärning, så de måste få reda på "utifrån" vilken tärning de ska använda:

Player

```
/** Skapar en spelare med namnet name */  
Player(String name);  
  
/** Spelaren kastar tärningen die tills det blir  
    två lika i följd, returnerar antalet kast */  
int play(Die die);  
  
/** Tar reda på spelarens namn */  
String getName();
```

Klassen DiceGame, en spelomgång, 1

```
public class DiceGame {  
    private Player player1;  
    private Player player2;  
    private Die die;  
  
    /** Skapar ett spel som spelas mellan spelarna  
        player1 och player2 */  
    public DiceGame(Player player1, Player player2) {  
        this.player1 = player1;  
        this.player2 = player2;  
        die = new Die();  
    }  
  
    /** Genomför en spelomgång, returnerar vinnaren */  
    public Player play() {  
        // ... nästa bild  
    }  
}
```

Klassen DiceGame, en spelomgång, 2

```
public class DiceGame {  
    ...  
  
    /** Genomför en spelomgång, returnerar vinnaren */  
    public Player play() {  
        int p1Rolls = player1.play(die);  
        int p2Rolls = player2.play(die);  
        while (p1Rolls == p2Rolls) {  
            p1Rolls = player1.play(die);  
            p2Rolls = player2.play(die);  
        }  
        return (p1Rolls < p2Rolls) ? player1 : player2; //villkors-uttryck  
    }  
}
```

Villkorsuttryck, ? :

```
logiskt uttryck ? uttryck1 : uttryck2
```

Exempel:

```
int i = 1;
int j = 2;
int result = (i == 1) ? i + 5 : j + 5; // result = 6
return (i > j) ? i : j;                // 2 returneras
```

Man kan klara sig utan villkorsuttryck:

```
int result;
if (i == 1) {
    result = i + 5;
} else {
    result = j + 5;
}
```

Implementation av Player.play

```
public class Player {  
    ...  
  
    /** Spelaren kastar tärningen die tills det blir  
        två lika i följd, returnerar antalet kast */  
    public int play(Die die) {  
        die.roll();  
        int prevResult = die.getResult();  
        die.roll();  
        int result = die.getResult();  
        int nbrRolls = 2;  
        while (result != prevResult) {  
            die.roll();  
            nbrRolls = nbrRolls + 1;  
            prevResult = result;  
            result = die.getResult();  
        }  
        return nbrRolls;  
    }  
}
```

Klassen DiceGame med do-while

```
public class DiceGame {  
    ...  
  
    /** Genomför en spelomgång, returnerar vinnaren */  
    public Player play() {  
        int p1Rolls;  
        int p2Rolls;  
        do {  
            p1Rolls = player1.play(die);  
            p2Rolls = player2.play(die);  
        } while (p1Rolls == p2Rolls)  
        return (p1Rolls < p2Rolls) ? player1 : player2;  
    }  
}
```

Formatering

Automatisk formatering vid utskrift:

```
int sum = 209;  
System.out.println("Summan är " + sum);
```

Formatering utan utskrift:

```
int sum = 209;  
String s1 = "" + sum;           // "209"  
String s2 = String.valueOf(sum); // "209"
```

Se även `System.out.printf`

Utskrift

Metoder i **PrintWriter** som kan användas på `System.out`:

PrintWriter

```
void print(String s);    // skriv strängen s
void println(String s); // "print line", som print men avsluta med övergång till ny rad
void println();         // enbart ny rad
void flush();           // skicka upplagrade utskrifter till skärmen
```

Formatering av utskrift med `printf`, se ankboken 7.9, s 110:

```
for (int i = 2; i <= 5; i++) {
    double r = Math.sqrt(i);
    System.out.printf("%5d...%6.3f%n", i, r);
}
```


Utskrift på fil

```
import java.util.Random;
import java.io.PrintWriter;
import java.io.File;
import java.io.FileNotFoundException;

public class TryCatchExample {
    public static void main(String[] args) {
        PrintWriter out = null;
        try {
            out = new PrintWriter(new File("random.txt"));
        } catch (FileNotFoundException e) {
            System.err.println("Filen random.txt kunde inte öppnas");
            System.exit(1);
        }
        //... utskrifter här med System.out.print hamnar på filen
    }
}
```

Switch-sats

I stället för en sekvens av if ... else if ... else if ... kan man använda en **switch**-sats. **Glöm inte break!**

```
switch (w.getKey()) {  
  case 'a':  
    turtle.rotate(5);  
    break;  
  case 's':  
    turtle.rotate(-5);  
    break;  
  case ' ':  
    turtle.forward(10);  
    break;  
  case 'r':  
    int someRandomX = (int) (Math.random() * 100 + 1);  
    int someRandomY = (int) (Math.random() * 100 + 1);  
    turtle.moveTo(someRandomX, someRandomY);  
    break;  
  default:  
    break;  
}
```

Se hela koden här i [SpriteTest.java](#)

Broken Switch

Det blir **tokigt** om man **glömmer break**:

```
1 package week05;
2 import java.util.Scanner;
3
4 public class BrokenSwitch {
5     public static void main(String[] args) {
6         System.out.println("Ange första bokstaven i din favoritgrönsak:");
7         Scanner scan = new Scanner(System.in);
8         String line = scan.nextLine();
9         char ch = line.length() > 0 ? line.charAt(0) : ' ';
10        switch (ch) {
11            case 'g':
12                System.out.println("gurka");
13            case 'z':
14                System.out.println("zucchini");
15                break;
16            case 't':
17                System.out.println("tomat");
18            case 'b':
19                System.out.println("broccoli");
20            default:
21                System.out.println("annan grönsak");
22                //break behövs inte i default om sist, men bra ändå vid framtida ändring
23        }
24    }
25 }
```