Kompilera och exekvera Javakod

Förberedelser

För att kunna göra dessa övningar måste du ha installerat Java Development Kit, JDK, som bland annat innehåller Java-kompilatorn, javac. Hur du installerar detta program ser på kurswebben. Eventuellt måste du också ändra systemvariabeln PATH, så att operativsystemet hittar kompilatorn och andra program som behövs[1].

Du behöver också en enkel texteditor att skriva källkoden i, Wordpad eller liknande duger.

I denna övning kommer kompilering och exekvering att göras från ett terminalfönster, t.ex. kommandotolken i Windows.

Källkod och bytekod, javatolken

Den källkod du skriver i Java kompileras inte till maskinkod utan till s.k. java-bytekod. Denna bytekod tolkas sedan vid exekveringen av ett annat program, javatolken (interpretatorn).

Kompilering sker i ett terminalfönster med kommandot javac, och exekvering med kommandot java (som startar javatolken).

Ett första enkelt exempel

Hämta filen `WelcomeToJava.java` från första föreläsningen på kurswebben, och spara den t.ex. i en katalog som du namnger Java på din hemvolym. Öppna filen i en enkel texteditor som WordPad och studera källkoden.

```
import java.util.Scanner;

public class WelcomeToJava {

	public static void main(String[] args) {

		Scanner scan = new Scanner(System.in);
		String name;

		System.out.print("Hi, what's your name? ");
		name = scan.next();

		System.out.println("Welcome to the Java world, " + name);
	}
}
```

Programmet läser in ett namn (typen String) och skriver ut en hälsning.

[1] På Windows gör du detta i Control Panel, System, Advanced och sedan Enviroment variabels. Lägg till, i slutet av system variabeln PATH, sökvägen till mappen bin i installationen av JDK, t.ex.
; C:\Program Files\Java\jdk1.6.0_20\bin

Kompilera

Öppna ett terminalfönster, t.ex. kommandotolken på Windows (Start, Run…, cmd) *och flytta dig till den katalog där du sparade filen*[2].

Kompilera källkoden med

```
> javac WelcomeToJava.java
```

Om du inte får några meddelanden gick kompileringen bra.

Lista innehållet i katalogen. Det ska nu finnas ytterligare en fil där, `WelcomeToJava.class`. Det är denna fil som innehåller bytekoden.

Exekvera

Du exekverar programmet genom att ange kommandot `java` tillsammans med bytekodfilens namn, *utan* ändelsen class. Detta startar javatolken som tolkar bytekoden och utför instruktionerna.

Så här kan det se ut

```
> java WelcomeToJava
Hi, what's your name? Anders
Welcome to the Java world, Anders
>
```

Felmeddelanden

Ändra i källkoden genom att kommentera bort raden där name deklareras (`//String name;`). Spara om filen och kompilera med `javac` igen.

Du får nu ett felmeddelande vid kompileringen, se nedan.

```
> javac WelcomeToJava.java
WelcomeToJava.java:11: cannot find symbol
symbol  : variable name
location: class WelcomeToJava
    name = scan.next();
    ^
WelcomeToJava.java:13: cannot find symbol
symbol  : variable name
location: class WelcomeToJava
    System.out.println("Welcome to the Java world, " + name);
                                                      ^
2 errors
```

Notera ut du får information om i vilken fil, på vilken rad och var på raden felet hittades och vad som orsakade felet (i detta fall har ju inte variabeln name deklarerats innan den användes, härav ”cannot find symbol, variable name”).

Ändra tillbaka och spara om filen.

---

[2] Använd `cd`, change directory, för att byta katalog (`cd ..` för att flytta upp en nivå) och `dir` eller `ls` för att lista innehållet i katalogen.

Viktig syntax

När du skriver egna Java program är följande viktigt

- Javakod skrivs alltid i en klass,
  `public class MyApplication { …`
- Namnet på filen och klassen måste matcha varandra exakt (även små och stora bokstäver). I exemplet ovan måste alltså filen döpas till *MyApplication.java*.
- Main-metoden anges alltid
  `public static void main(String[] args) { …`

Övningsuppgifter

Källkoden till följande uppgifter kompileras och exekveras med javac och java i ett terminalfönster. När du skriver källkoden är det enklast att utgå från den fil du redan har (WelcomeToJava.java) och spara om den i ett nytt namn. Glöm inte att du måste ändra klassnamnet så att det matchar det nya filnamnet.

Uppgift 1

En bank tillämpar följande räntesats på ett sparkonto.
För den del av beloppet som understiger 100 000 kr är räntan 0.5%
För den del av beloppet som överstiger 100 000 kr är räntan 2.5%
Skriv ett program som räknar ut årsräntan om man har k kr på kontot under hela året.

Använd flyttalstypen double för beloppen. Du läser in ett flyttal från tangentbordet med

```
double x;
x = scan.nextDouble();
```

Uppgift 2

Skriv ett program som läser in ett godtyckligt antal heltal i en array och sedan skriver ut dessa i omvänd ordning.

Du skapar arrayen med

```
int[] numbers = new int[n];
```

Du läser in heltalen med

```
for(int i = 0; i < numbers.length; i++) {
    numbers[i] = scan.nextInt();
}
```

Uppgift 3

Skriv ett program som slumpar 1000 tärningskast och skriver ut frekvenserna för respektive resultat (d v s antal 1:or, antal 2:or, o s v).
Till din hjälp behöver du en array med plats för 6 heltal, frekvenserna .

Du slumpar ett tal i intervallet 1-6 med

```
int x;
x = (int) (Math.random()*6 + 1);
```

Uppgift 4

Burrleken går till på följande sätt. Man bestämmer ett burrtal mellan 2 och 9. Talen 1 t o m 99 skrivs ut, i tur och ordning, men de tal som är jämnt delbara med burrtalet eller innehåller burrtalet som en siffra ersätts med ordet ”burr”. Skriv ett program som leker burrleken, t ex med talet 3.

```
Burrtalet: 3
1 2 burr 4 5 burr 7 8 burr 10
11 burr burr 14 ...
```

Använd heltalstypen `int`. Rest vid heltalsdivision får du med operatorn %. Talet n är jämt delbart med 3 om `n%3 == 0`.

Appendix

Att skicka argument till main

I Java krävs att metoden main() har en parameterlista ,

```
public static void main(String[] argv)
```

Du kan alltså skicka en array med textsträngar till programmet som kan användas för att ge indata eller styra programmets exekvering[3]. Studera följande program:

```
public class Eko {

    public static void main (String[] args) {

        for (int i = 0; i < args.length; i++)
            System.out.println(args[i]);
    }
}
```

Om programmet kompileras och sedan exekveras med:

```
> java Eko Anders Andersson
```

fås utskriften:

```
Anders
Andersson
```

Om programmet behöver numeriska indata måste den String som ges till programmet omvandlas till ett tal på t ex detta sätt:

```
int x;
if (args.length > 0)
          x = Integer.parseInt(args[0]);
```

Exekveringsfel

Program som inte innehåller syntaxfel och alltså kompileras till en körbar klass kan ändå generera fel vid körningen, t ex om programmet indexerar utanför en array. Denna typ av fel kallas Exceptions (i det aktuella fallet ArrayIndexOutOfBoundsException).

Vid exekveringsfel meddelar Javatolken i vilken klass, metod och på vilken rad i källkoden felet uppstod, vilket underlättar felsökandet.

[3] Du har faktiskt samma möjlighet i C-program, men i Java är argumentlistan ett krav.

Vanliga felmeddelanden vid exekvering

Om du vid exekvering får felmeddelandet

```
Exception in thread "main" java.lang.NoClassDefFoundError:
WelcomeToJava /class
```

har du förmodligen, vid exekveringen, angett

```
> java WelcomeToJava.class
```

Argumentet till javatolken är klassnamnet *utan* ändelsen .class.

Om du försöker exekvera en klass som saknar main()-metod får du följande felmeddelande:

```
> java ProgramExempel
Exception in thread "main" java.lang.NoSuchMethodError: main
```

Notera också att du måste skilja på stora och små bokstäver när du anger filnamn och klassnamn.

Just-In-Time-kompilering

För att öka exekveringshastigheten används bl a *just-in-time*-kompilering vilket innebär att interpretatorn under exekveringen skapar verklig maskinkod för kodstycken som exekverats många gånger. Maskinkoden används sedan när de aktuella instruktionerna upprepas senare.

Om du testkör ett program som ger exekveringsfel kan *just-in-time*-kompileringen göra att du inte får information om var felet uppstod. Detta undviker du genom att vid exekveringen ange:

`java –Djava.compiler=NONE Klassnamn`, viket stänger av *just-in-time*-kompileringen.
När programmet är färdigtestat och felfritt kör du det på vanligt sätt för att öka exekveringshastigheten.