# Ohjelmoinnin perusteet R0027

# E
# Toistolauseet (for)

**Viopen luku 6 loppuun**

# for - syntaksi

```
for(alkutoimet; toistoehto; lopputoimet) {
   lauseita;
}
```

- **Alkutoimet**: Laskurin määrittely ja alkuarvon asettaminen
- **Toistoehto**: Laskurin arvon vertaaminen loppuarvoon
- **Lopputoimet**: Laskurin arvon askeltaminen (yleensä kasvattaminen yhdellä)

# for - toiminta

```
for(int i=1; i<=10; i++) {
  System.out.println(i + " potenssiin 2 on " +i*i);
}
```

```
System.out.println("Lasketaan nollasta ysiin");

for(int i=0; i<10; i++) {
  System.out.println("Kierros" +i);
}
```

# Sisäkkäiset silmukat

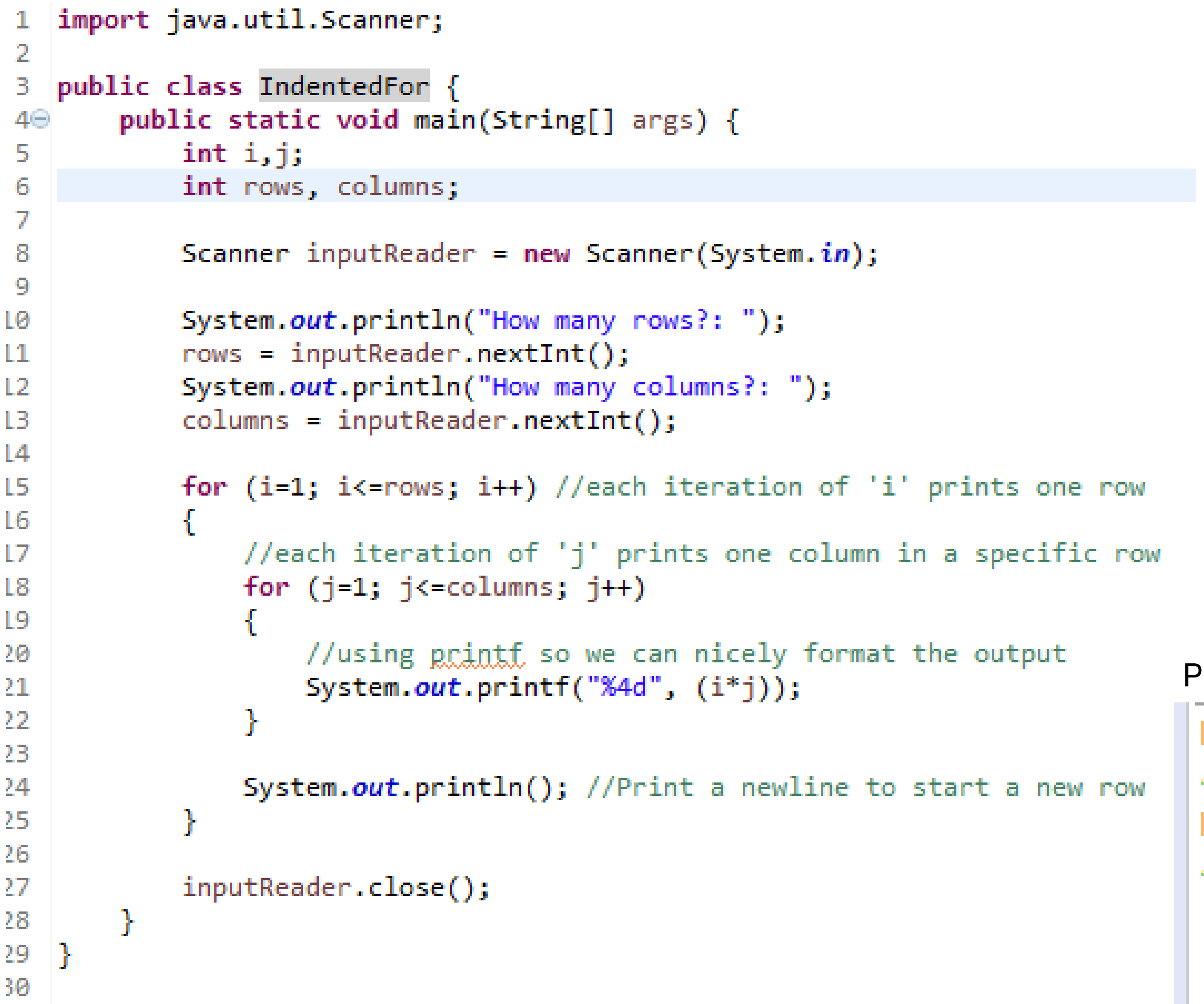
- Silmukoita voidaan tehdä myös sisäkkäin.
- Tällöin ulomman silmukan kullakin yksittäisellä kierroksella suoritetaan sisemmän silmukan kaikki kierrokset läpi

# Sisäkkäiset silmukat -esimerkki

- Kysy käyttäjältä kuinka monta riviä halutaan tulostaa
- Kysy käyttäjältä kuinka monta saraketta halutaan tulostaa
- Tulosta ruudulle kertotaulu ohessa esitetyllä tyylillä

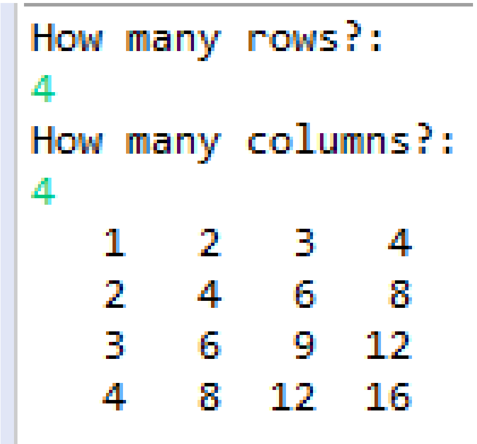
| | | | |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| 2 | 4 | 6 | 8 |
| 3 | 6 | 9 | 12 |
| 4 | 8 | 12 | 16 |

# Solution: (check file IndentedFor.java in Optima)



```java
import java.util.Scanner;

public class IndentedFor {
    public static void main(String[] args) {
        int i,j;
        int rows, columns;

        Scanner inputReader = new Scanner(System.in);

        System.out.println("How many rows?: ");
        rows = inputReader.nextInt();
        System.out.println("How many columns?: ");
        columns = inputReader.nextInt();

        for (i=1; i<=rows; i++) //each iteration of 'i' prints one row
        {
            //each iteration of 'j' prints one column in a specific row
            for (j=1; j<=columns; j++)
            {
                //using printf so we can nicely format the output
                System.out.printf("%4d", (i*j));
            }

            System.out.println(); //Print a newline to start a new row
        }

        inputReader.close();
    }
}
```

Program output:



```
How many rows?:
4
How many columns?:
4
   1   2   3   4
   2   4   6   8
   3   6   9  12
   4   8  12  16
```

# Debugging code with Eclipse – getting started

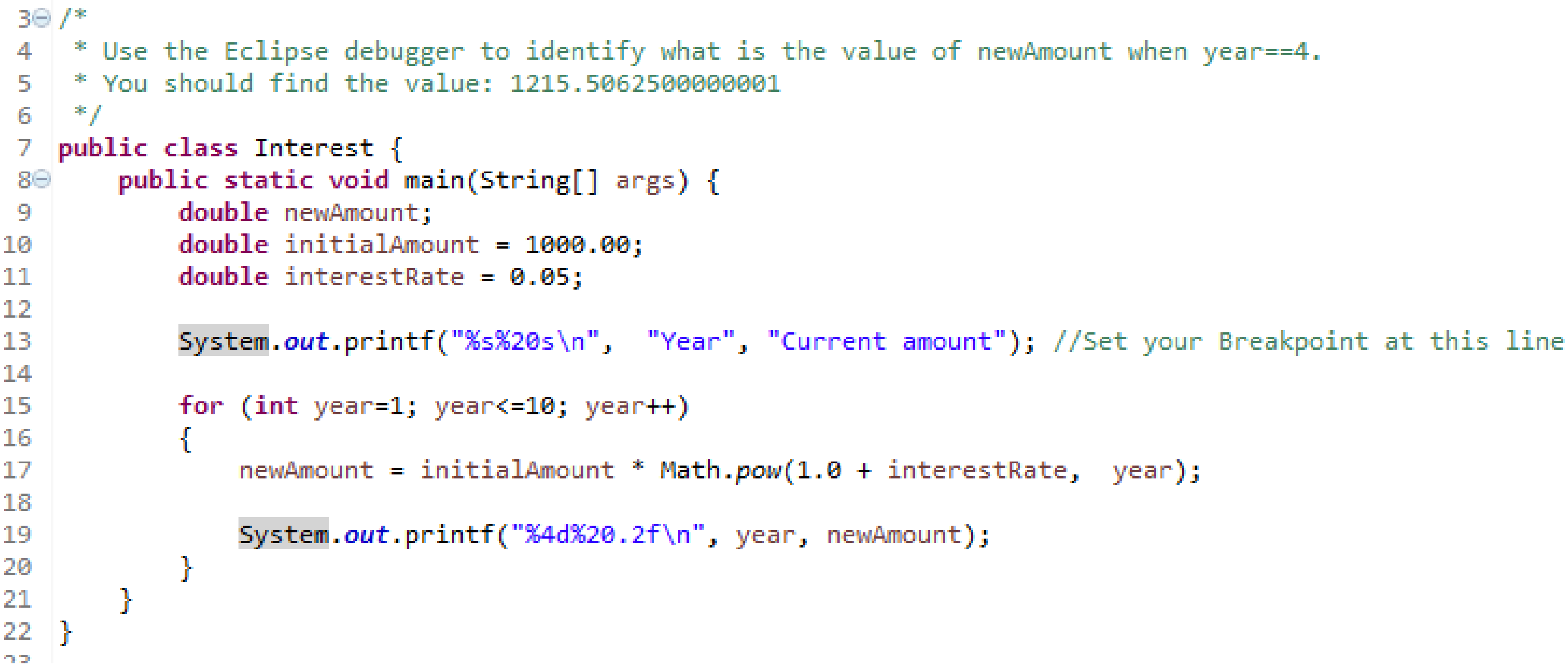
- Perspectives
  - Java perspective, the one you have been using
  - Debug perspective, with useful views to debug code, inspect variable values, and step through the code
- Breakpoint
  - suspends the execution of a program at the location where the breakpoint is set
- Step over
  - step over a method call (without entering it) at the currently executing line of code
- Step into
  - step into the next method call at the currently executing line of code
- Resume
  - Continue program execution until the next Breakpoint

# Debugging code with Eclipse – some resources

- Java Debugging with Eclipse – Tutorial
  - http://www.vogella.com/tutorials/EclipseDebugging/article.html
- Eclipse – Debugging a program (tutorial)
  - https://www.tutorialspoint.com/eclipse/eclipse_debugging_program.htm

# Exercise

Get the file Interest.java from Optima and debug it according to the instructions written in it comments



```java
/*
 * Use the Eclipse debugger to identify what is the value of newAmount when year==4.
 * You should find the value: 1215.5062500000001
 */
public class Interest {
    public static void main(String[] args) {
        double newAmount;
        double initialAmount = 1000.00;
        double interestRate = 0.05;

        System.out.printf("%s%20s\n",  "Year", "Current amount"); //Set your Breakpoint at this line

        for (int year=1; year<=10; year++)
        {
            newAmount = initialAmount * Math.pow(1.0 + interestRate,  year);

            System.out.printf("%4d%20.2f\n", year, newAmount);
        }
    }
}
```

# Tulostuksen muotoilu: printf

- System.out.printf(muotoilu, muuttuja);
- Muotoilun kaava: %[flags][width][.precision]converter

Examples of possible values:

| Flag | Meaning |
|---|---|
| **-** | Left justify. Default: right-justify |
| **+** | Output the signal (+/-) of a numerical value |
| **0** | Zero-pad numerical values |

| Converter | Meaning |
|---|---|
| **d** | Decimal integer (int) |
| **f** | Floating-point number (double \| float) |
| **c** | Character (char) |
| **s** | String |

Esim.

- %5d - kokonaisluku 5 merkin kentässä
- %.2f - kahden desimaalin tarkkuus

# Printf (esimerkkejä)

```
Rahan arvo = 66.66666666666667

System.out.printf("(%.2f)", raha);
(66,67)
System.out.printf("(%-16.2f)", raha);
(66,67           )
System.out.printf("(%16.3f)", raha);
(          66,667)
System.out.printf("(%016.5f)", raha);
 (0000000066,66667)
System.out.printf("(%+16.3f)", raha);
(         +66,667)
System.out.printf("(%+16.3f)", raha*-1);
(         -66,667)
```

# Printf (other references)

- Check also this easy to understand summary:

- https://alvinalexander.com/programming/printf-format-cheat-sheet