

Systematic Program Development

Wolfgang Schreiner

Research Institute for Symbolic Computation (RISC)

Johannes Kepler University, Linz, Austria

Wolfgang.Schreiner@risc.jku.at

<http://www.risc.jku.at>

Printing Stars

Problem Statement

Print n lines of output of form

```
*  
**  
***  
****  
*****  
...
```

- i -th line shall contain i stars
- n is provided by the user as text input.

Program Structure

```
public static void main(String[] args)
{
    int n = 0; // number of lines

    // read number n from user
    ...

    // print n lines
    ...
}
```

Two problems to be solved.

Input Problem

Read number n from user.

```
// read n, exits on empty input, retries on invalid input
while (true)
{
    System.out.print("Enter number of lines: ");
    n = Input.readInt();
    if (Input.isOkay()) break;
    if (Input.hasEnded())
    {
        System.out.println("No input given!");
        System.exit(-1);
    }
    System.out.println("Invalid input!");
    Input.clearError();
}
```

Printing Problem

Print n lines.

- Problem is of iterative nature:
 - Print one line after the other.

```
// print n lines
for (int i = 1; i <= n; i++)
    print line with i stars
```

Problem is reduced to a simpler subproblem.

Subproblem

Print line with i stars.

- Problem is of composed nature:
 - Print one star after the other.
 - Finally terminate line.

```
// print line with i stars
for (int j = 1; j <= i; j++)
    System.out.print("*");
System.out.println("");
```

Subproblem can be solved directly.

Printing Program

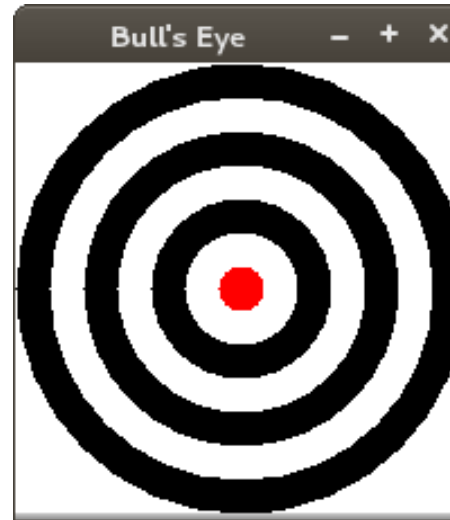
Insert solution of subproblem.

```
// print n lines
for (int i = 1; i <= n; i++)
{
    // print line with i stars
    for (int j = 1; j <= i; j++)
        System.out.print("*");
    System.out.println("");
}
```

All problems have been solved.

Drawing Circles

Problem Statement



- Draw a “bull’s eye” :
 - Alternating circles of black and white of particular width.
 - Innermost circle in red.
- Parameters: disk RADIUS and circle WIDTH.

General Program Structure

```
import kwm.*;
import java.awt.*;

public class BullsEye {
    public static void main(String[] args) {
        final int RADIUS = 100;    // disk radius
        final int WIDTH = 15;      // width of each circle drawn
        Drawing.begin("Bull's Eye", 2*RADIUS, 2*RADIUS);

        // draw bull's eye
        ...

        Drawing.end();
    }
}
```

Solution Strategy

Need an algorithm for solving the problem.

- Graphical primitive: `fillOval`.
 - Paint a filled ellipse (circular disk).
- Idea: paint one disk on top of each other.
 - Outermost disk in black.
 - Smaller disk in white.
 - ...
 - Smallest disk in red.

We will elaborate this idea to an algorithm/program.

Core Program Structure

```
// draw alternating circles of black and white
// starting with outermost circle with radius = RADIUS
// iteratively shrink radius by WIDTH until radius <= WIDTH
for (int radius = RADIUS; radius > WIDTH; radius -= WIDTH)
{
    switch color from black to white or from white to black
    draw circle with denoted radius and denoted color
}
draw innermost circle in red
```

- Switching the color from black to white and vice versa.
 - Need a boolean flag which indicates whether the next disk is black or white.
 - In each iteration, the value of the flag is inverted.

Need to refine the core program structure.

Refined Core Program Structure

```
boolean black = true;
for (int radius = RADIUS; radius > WIDTH; radius -= WIDTH)
{
    // set color, switch to other color
    if (black)
    {
        Drawing.graphics.setColor(Color.black);
        black = false;
    }
    else
    {
        Drawing.graphics.setColor(Color.white);
        black = true;
    }
    draw circle with denoted radius and denoted color
}
draw the innermost circle in red
```

Drawing a Circle

Draw circle with denoted radius and denoted color.

```
// draw circle of denoted radius with center (MID_X, MID_Y)
Drawing.graphics.fillOval(MID_X-radius, MID_Y-radius, 2*radius, 2*radius);
```

- Center coordinates MID_X and MID_Y.

- Forgotten in our original problem statement.
- Have to add them to the set of parameters.

```
final int RADIUS = ...; // disk radius
final int WIDTH = ...; // width of each circle drawn
final int MID_X = RADIUS; // x coordinate of center
final int MID_Y = RADIUS; // y coordinate of center
```

Drawing the Innermost Circle

Draw the innermost circle in red.

- Know how to draw the circle.
- Know the center coordinates `MID_X` and `MID_Y`.
- What is the radius of the circle?
 - Example: `WIDTH=15`; in last iteration *radius*=25.
 - Next update: *radius*=10; loop terminates.
 - Radius of red circle is 10.
- Problem: variable is declared local to the loop:

```
for (int radius = ...; ...; ...)
```

 - Final value is not visible outside of loop.

How to Determine the Innermost Radius?

1. Compute the final radius from RADIUS and WIDTH.

- How?

2. Place code inside the loop:

```
for (int radius = RADIUS; radius > WIDTH; radius -= WIDTH)
{
    ...
    if (radius-WIDTH <= WIDTH)
        draw circle of radius "radius-WIDTH" in red
}
```

3. Draw variable declaration out of loop:

```
int radius = 0;
for (radius = RADIUS; radius > WIDTH; radius -= WIDTH)
    ...
    draw circle of denoted radius in red
```

Drawing the Innermost Circle

Have the radius of the innermost circle

```
// draw red center circle  
Drawing.graphics.setColor(Color.red);  
Drawing.graphics.fillOval(MID_X-radius, MID_Y-radius, 2*radius, 2*radius);
```

Have solved all subproblems.

Analyzing Exam Grades

Problem Statement

- Analyze exam grades (1–5).
- Grades are read from the standard input stream.
- Determine the best grade, the worst grade, the average grade.
- Print results to standard output stream.
- Program shall process the grades of multiple exams.

General Program Structure

```
// analyze exam grades
char answer = 0;
do
{
    analyze exam;
    print result;
    ask user whether to continue;
}
while (answer == 'y');
```

- Iterative problem:
 - Analyze one exam.
 - Print the result.
 - Ask the user whether to continue.

Ask User Whether to Continue

```
// ask user whether to continue
while(true)
{
    System.out.print("Another exam (y/n)? ");
    answer = Input.readChar();
    if (Input.isOkay() && (answer == 'y' || answer == 'n')) break;
    if (Input.hasEnded())
    {
        System.out.println("Unexpected end of input!");
        System.exit(-1);
    }
    System.out.println("Invalid input!");
    Input.clearError();
}
```

Variable “answer” set to result.

Analyze Exam

```
// analyze exam
int count = 0;           // number of grades processed so far
int sum = 0;            // sum of grades processed so far
int min = Integer.MAX_VALUE; // minimum grade processed so far
int max = Integer.MIN_VALUE; // maximum grade processed so far
read and process exam
```

- Need two values to compute the average:
 - Number of grades, sum of grades.
- Initialization of min and max:
 - First input will set min and max to new value.

Print Result

```
// print result
System.out.println("\nNumber of grades: " + count);
System.out.println("Best grade: " + min);
System.out.println("Worst grade: " + max);
System.out.println("Average grade: " + sum/count);
```

Based on minimum, maximum, sum, and count.

Read and Process Exam

We have to determine the end of an exam.

```
// read and process exam
while (true)
{
    int grade = 0;
    read grade
    if (grade == 0) break;
    process grade
}
```

We use “sentinel” grade 0 to denote the end of an exam.

Read Grade

```
// read grade
while(true)
{
    System.out.print("Enter grade " + (count+1) + " (0 when done): ");
    grade = Input.readInt();
    if (Input.isOkay() && 0 <= grade && grade <= 5) break;
    if (Input.hasEnded())
    {
        System.out.println("Unexpected end of input!");
        System.exit(-1);
    }
    System.out.println("Invalid input!");
    Input.clearError();
}
```

We have to perform sanity checks.

Process Grade

```
// process grade  
count++;  
sum += grade;  
if (grade < min) min = grade;  
if (grade > max) max = grade;
```

Now the initialization of `min` and `max` makes sense.

Abstraction Levels

Gradual reduction of problems.

analyze exam grades
analyze exam
read and process exam
read grade
process grade
print result
ask user whether to continue

Program structure reflects refinements.

Program Structure

```
char answer = 0;
do
{
    // analyze exam
    ...
    // read and process exam
    while (true)
    {
        int grade = 0;
        read grade
        if (grade == 0) break;
        process grade
    }
    print result
    ask user whether to continue
}
while (answer == 'y');
```

Testing the Program

```
Enter grade 1 (0 when done): 3
Enter grade 2 (0 when done): 4
Enter grade 3 (0 when done): 5
Enter grade 4 (0 when done): 0
```

```
Number of grades: 3
Best grade: 3
Worst grade: 5
Average grade: 4
```

Looks fine.

Testing the Program

```
Another exam (y/n)? y  
Enter grade 1 (0 when done): 2  
Enter grade 2 (0 when done): 4  
Enter grade 3 (0 when done): 4  
Enter grade 4 (0 when done): 0
```

```
Number of grades: 3  
Best grade: 2  
Worst grade: 4  
Average grade: 3
```

Just great.

Testing the Program

```
Another exam (y/n)? y
```

```
Enter grade 1 (0 when done): 0
```

```
Number of grades: 0
```

```
Best grade: 2147483647
```

```
Worst grade: -2147483648
```

```
Exception in thread "main" java.lang.ArithmeticException: / by zero  
    at Exams.main(Exams.java:51)
```

We forgot the case that an exam may have zero grades!

Revised Print Result

```
// print result
System.out.println("\nNumber of grades: " + count);
if (count > 0)
{
    System.out.println("Best grade: " + min);
    System.out.println("Worst grade: " + max);
    System.out.println("Average grade: " + sum/count);
}
```

Analysis only makes sense if there is at least one grade.

Testing the Program

Enter grade 1 (0 when done): 0

Number of grades: 0

Another exam (y/n)?

We fixed the bug.

Testing the Program

```
Another exam (y/n)? y
Enter grade 1 (0 when done): 2
Enter grade 2 (0 when done): 4
Enter grade 3 (0 when done): 4
Enter grade 4 (0 when done): 5
Enter grade 5 (0 when done): 0
```

```
Number of grades: 4
Best grade: 2
Worst grade: 5
Average grade: 3
```

Strange average.

Computing the Average

We examine the computation of the average:

- Our version uses integer division:

```
System.out.println("Average grade: " + sum/count);
```

- Better use floating point division:

```
System.out.println("Average grade: " + (float)sum/(float)count);
```

– We can write this in a somewhat simpler way (how?).

When computing with numbers, we have to take care of the datatypes.

Testing the Program

```
Enter grade 1 (0 when done): 2  
Enter grade 2 (0 when done): 4  
Enter grade 3 (0 when done): 4  
Enter grade 4 (0 when done): 5  
Enter grade 5 (0 when done): 0
```

```
Number of grades: 4  
Best grade: 2  
Worst grade: 5  
Average grade: 3.75
```

We have also solved this problem.

Stepwise Refinement

- Solution process:
 - Problem is decomposed into simpler subproblems.
 - Subproblems are decomposed again.
 - Ultimately, problems are simple enough to be solved directly.
- Implementation:
 - Structure of program reflects problem decomposition.
 - Problem: main method becomes big monolithic piece of code.

Bigger programs are decomposed into separate methods.