CSE2016 Programming Methodology

Week 7: Patterns of Repetition: Iteration and Recursion

Instructor: Jinyoung Han (jinyounghan@hanyang.ac.kr)

HANYANG UNIVERSITY

Contents

HANYANG UNIVERSITY

Today's Schedule

01. Repetition

Repetition

- Some jobs must be solved by repeating some step over and over

- Example

```
public static void main(String[] args)
{
  System.out.println("1/2 = " + (1.0/2));
  System.out.println("1/3 = " + (1.0/3));
  System.out.println("1/4 = " + (1.0/4));
  ...
  System.out.println("1/20 = " + (1.0/20));
}
```

Can we do until 1/10000, by hand?

02. While Loops

While Loops

- Format in Java

```
while (Test) { Body }
```

- Semantics
  - 1. Calculate the question/condition.
  - 2. If it is true, “Body”. Go to #1.
  - 3. Otherwise, the loop ends.

03. Definite Iteration

Loops

- Definite Iteration
  - A loop performs “definite iteration” when the number of the loop's iterations is known at the moment the loop is started
  - cf) indefinite iteration

- Unbounded iteration
  - Iterating infinitely
  - a.k.a., “Loop is diverged”

03. Definite Iteration

computeAverage

- Goal: computing an average
  - average = (exam1 + … + examN) / N

- Algorithm
  - Assumption: we know the number of exams, N
  - sum = 0, count = 0
  - Loop (count != N)
    - Get score for an exam
    - sum = sum + score
    - count = count + 1
  - return sum / N

03. Definite Iteration

computeAverage

```
public class computeAverage
{
    public static void main(String[] args)
    {
        double total_points = 0.0;
        int count = 0;
        int how_many = 5;
        while (count != how_many)
        {
            String input = JOptionPane.showInputDialog("Type next exam
            score:");
            int score = new Integer(input).intValue();
            total_points = total_points + score;
            count = count + 1;
        }
        JOptionPane.showMessageDialog(null, "Average score:" +
        total_points / how_many);
    }
}
```

Loop Invariant

- Loop invariant
  - A property of a loop that is true before (and after) each iteration

- Can be proved by “proof by induction”
  - Basic step
    - We show the invariant is true at the very first time the loop is encountered
  - Inductive step
    - We assume the invariant is already holding true at the start of an arbitrary iteration of the loop, and we show that when the iteration completes the invariant is still true

03. Definite Iteration

HANYANG UNIVERSITY

A Pattern

- A pattern for definite iteration
  - Save the loop counter
  - Ask whether the loop counter holds the condition
  - Increase the loop count in the body

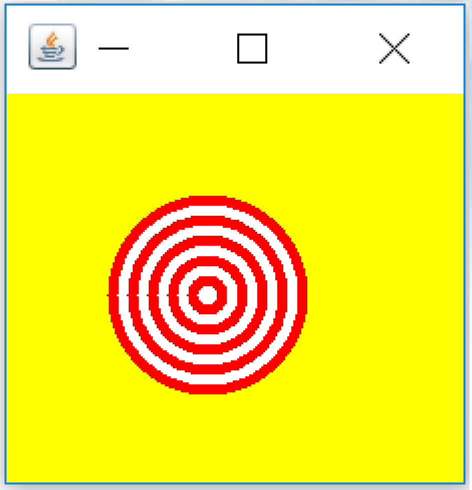
- A pattern

```
int count = initial value;
while (count != condition)
{
            body
            count = count + 1;
}
```

03. Definite Iteration

BullsEye

- Draw a target with n circles
- When "n" is given, the number of loops is determined
- Necessary variables
  - Count
  - Color
  - Diameter



03. Definite Iteration

HANYANG UNIVERSITY

BullsEye

```
public void paintBullsEye(int x, int y, int rings, int size, Graphics g)
{
    int count = 0;
    int diameter = size;
    int width = size / rings;
    Color color = Color.red;

    while (count != rings)
    {
        int new_x = x + ((width * count)/2);
        int new_y = y + ((width * count)/2);
        g.setColor(color);
        g.fillOval(new_x, new_y, diameter, diameter);
        count = count + 1;
        diameter = diameter - width;
        if (color == Color.red)
                color=Color.white;
        else color=Color.red;
    }
}
```

03. Definite Iteration

Non-termination

- What if how_many = -1?

```
public class computeAverage
{
    public static void main(String[] args)
    {
        double total_points = 0.0;
        int count = 0;
        int how_many = -1;
        while (count != how_many)
        {
            String input = JOptionPane.showInputDialog("Type next exam score:");
            int score = new Integer(input).intValue();
            total_points = total_points + score;
            count = count + 1;
        }
        JOptionPane.showMessageDialog(null, "Average score:" + total_points /
        how_many);
    }
}
```

03. Definite Iteration

Remedy

- What if how_many = -1?

```
public class computeAverage
{
    public static void main(String[] args)
    {
        double total_points = 0.0;
        int count = 0;
        int how_many = -1;
        while (count < how_many)
        {
            String input = JOptionPane.showInputDialog("Type next exam score:");
            int score = new Integer(input).intValue();
            total_points = total_points + score;
            count = count + 1;
        }
        JOptionPane.showMessageDialog(null, "Average score:" + total_points /
        how_many);
    }
}
```

04. Indefinite Iteration

Input Processing

- When multiple inputs are given from a user, it may be convenient not to decide the number of inputs but to know during the input process
  - Using “cancel” or “end” button

- A pattern

```
boolean processing = true;
while (processing)
{
        get an input
        if (end?)
                processing = false;
        else
                { body }
}
```

04. Indefinite Iteration

Example

```
public static void main(String[] args)
{
     double total_points = 0.0;
     int count = 0;
     boolean processing = true;

     while (processing)
     {
          String input = JOptionPane.showInputDialog("Type next exam score or (precss
          Cancel to quit):");
          if (input == null)
                    processing = false;
          else
          {
               int score = new Integer(input).intValue();
               total_points = total_points + score;
               count = count + 1;
          }
     }
     if (count == 0)
          throw new RuntimeException("error: no input supplied");
     JOptionPane.showMessageDialog(null, "Average score:" + total_points / count);
}
```

04. Indefinite Iteration

Search

- Searching is a process of repetitions until the target value is found or the end

- A pattern

```
boolean item_found = false;
while (!item_found && remaining_items)
{
        examine the current item
        if (found)
                item_found = true;
        else
                move to the next item;
}
```

04. Indefinite Iteration

findChar

- Input: String S, character c
- Output: location of c in S

- Algorithm
    - index = 0;
    - Loop(index < S's length)
        - if S.charAt(index) == c
            - End
        - Otherwise
            - index = index + 1

04. Indefinite Iteration

findChar

```
public static void main(String[] args)
{
    String s = "How are you? I am fine thank you and you? ok";
    char c = 'p';

    boolean found = false;
    int index = 0;

    while (!found && index<s.length())
    {
        if (s.charAt(index) == c)
                found = true;
        else
                index = index + 1;
    }
    if (!found)
        System.out.println("not found.");
    else
        System.out.println("index: " + index);
}
```

04. Indefinite Iteration

isPrime

- Input: int n
- Output: validate whether n is prime or not

- Algorithm
    - current = n/2
    - Loop(current >= 2)
        - if (n%current == 0)
            - End
        - Otherwise
            - current = current - 1

04. Indefinite Iteration

HANYANG UNIVERSITY

isPrime

```
public class isPrime
{
     public static void main(String[] args)
     {
          int n = 6;
          if (n < 2)
                    throw new RuntimeException("error: invalid " + n );
          else
          {
                    boolean found = false;
                    int c = n/2;
                    while (!found && c > 1)
                    {
                         if (n%c == 0)
                              found = true;
                         else
                              c = c-1;
                    }
                    if (!found) System.out.println("yes.");
                    else System.out.println("no.");
          }
     }
}
```

05. For-Statements

For-Statements

- A widely used pattern
  - initialize the variable
  - for each iteration, ask about the variable
  - change the variable for each iteration

- A terse way
  - for (initialization; question; change) {body}

05. For-Statements

HANYANG UNIVERSITY

findChar

```
boolean found = false;
int index = 0;
while (!found && index<s.length())
{
        if (s.charAt(index) == c)
                found = true;
        else
                index = index + 1;
}
if (!found)
        System.out.println("not found.");
else
        System.out.println("index: " + index);
```

```
int index;
for (index = 0; index < s.length() && s.charAt(index) != c; index++);

if (index == s.length())
        System.out.println("not found.");
else
        System.out.println("index: " + index);
```

05. For-Statements

isPrime

```
boolean found = false;
int c = n/2;
while (!found && c > 1)
{
     if (n%c == 0)
          found = true;
     else
          c = c-1;
}
if (!found)
          System.out.println("yes.");
else
          System.out.println("no.");
```

⇩

```
int c;
for (c = n/2; c > 1 && n%c !=0; c = c - 1);
if (c == 1)
          System.out.println("yes.");
else
          System.out.println("no.");
```

06. Nested Loops

Nested Loops

- Loops inside a Loop
  - So far, we play at an 1-D plane
  - Let’s move on 2-D or m-D planes
    - With nested loops

06. Nested Loops

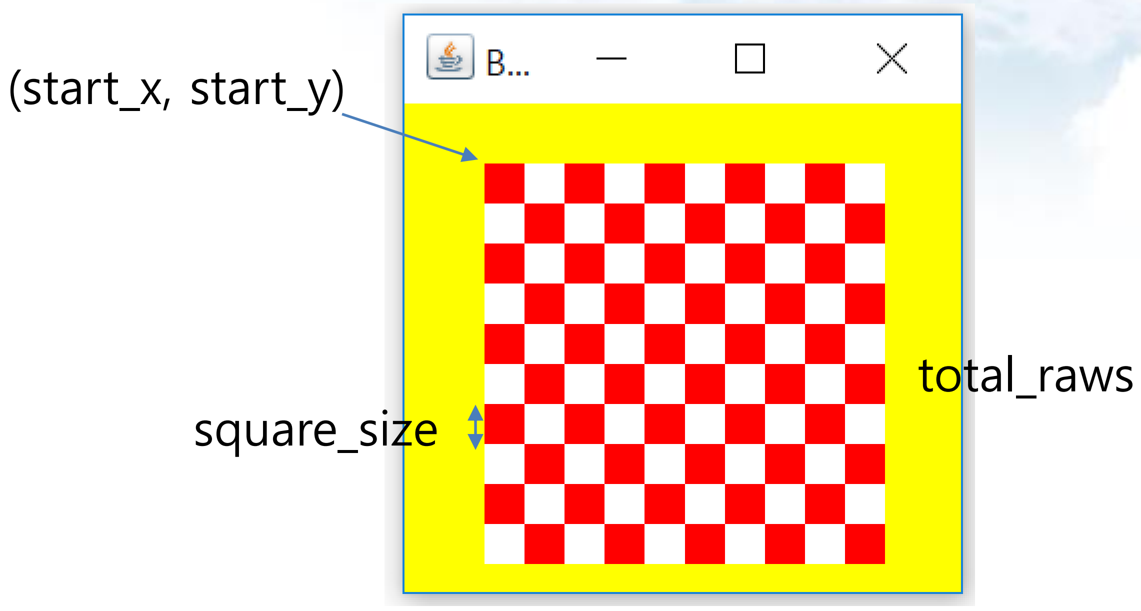
HANYANG UNIVERSITY

Multiplication

```java
public class Multiplication
{
    public static void main(String[] args)
    {
        for (int i=1; i < 10; i=i+1)
        {
            for (int j=1; j < 10; j=j+1)
            {
                System.out.print(i + "*" + j + "=" + (i*j) + " ");
            }
            System.out.println();
        }
    }
}
```

06. Nested Loops

Board



06. Nested Loops

Board

```
private void paintBoard(int start_x, int start_y, int total_rows, int
square_size, Graphics g)
{
    for (int x = 0; x < total_rows; x++)
    {
        int x_position = start_x + (x * square_size);
        for (int y = 0; y < total_rows; y++ )
        {
            int y_position = start_y + (y * square_size);
            if ( ((x + y) % 2) == 0 )
                g.setColor(Color.red);
            else
                g.setColor(Color.white);
            g.fillRect (x_position, y_position, square_size,square_size);
        }
    }
}
```

07. Recursion

Factorial

- A problem: calculate n!

- Two ways
  - n! = 1 * 2 * 3 * … * (n-1) * n
  - 0! = 1

    n! = n * (n-1)! if n > 0

- The second way
  - n! -> n * (n-1)! -> n * (n-1) * (n-2)! -> … -> n * (n-1) * … * 2 * 1

07. Recursion

HANYANG UNIVERSITY

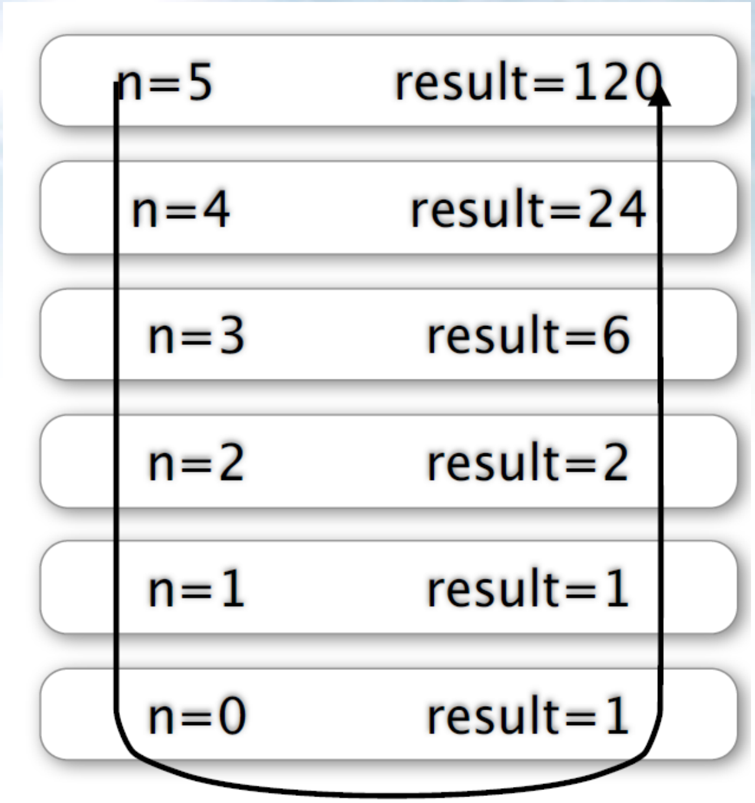
Mathematical Induction

- We want to calculate P(n) for all n

- Prove for P(0)
- Assumption
  - i < k for all i, P(i) is true
  - Prove P(k)
- By mathematical induction P(n) is true for all n

Factorial

```
public static long fac_recursion(int n)
{
        if(n==0)
                return 1;
        else
                return (n * fac_recursion(n-1));
}
```



07. Recursion

Factorial

```
public static long fac_recursion(int n)
{
		if(n==0)
			return 1;
		else
			return (n * fac_recursion(n-1));
}
```

Simple to think

```
public static long fac_loop(int n)
{
		long answer = 1;
		for(int i=1; i<=n; i++)
			answer = answer * i;
		return answer;
}
```

fast

07. Recursion

HANYANG UNIVERSITY

Fibonacci Numbers

- Definition
  - fib(0) = fib(1) = 1
  - fib(n) = fib(n-1) + fib(n-2)

n	F_n
0	1
1	1
2	2
3	3
4	5
5	8
6	13
7	21
8	34
9	55
10	89
11	144
12	233
13	377
14	610
15	987
16	1597
17	2584
18	4181
19	6765
20	10946

07. Recursion

Fibonacci Numbers

```
public static long fib(int n)
{
        if (n == 0 || n == 1)
                return 1;
        else
                return (fib(n-1) + fib(n-2));
}
```

08. Summary

Summary

- Iteration
  - while (condition) { body }
  - for (initial; condition; change) { body }

- Loop invariants
- Recursion

Thanks

Week 7: Patterns of Repetition: Iteration and Recursion
Instructor: Jinyoung Han (jinyounghan@hanyang.ac.kr)