CSE 21
Intro to Computing II

Lecture 4 – Methods(3)

Announcement

- Lab 3 due before start of next lab
  - Type your answers in a text file and submit it as an attachment
- Reading assignment
  - Chapter 7.1 to 7.4 of textbook

PEER ASSISTED LEARNING SUPPORT

Mon: 3-5pm at SSB 330
Tue: 6-7pm at SSB 330
Thu: 10am-12pm at SSB 320

- Go to **learning.ucmerced.edu**
- Click on **"Programs"**
- Scroll down and click on **Peer Assisted Learning Support (PALS) to find out more**
- Click on the **"Learning Support Schedule"**

OR

use this shortcut to go straight to the schedule:

http://bit.ly/PALS_Schedule

"Peer Assisted Learning Support,
Your learning community."

Method overloading

```
public static int getAmount(Scanner input, String name) { // 1
    System.out.print("Enter the amount of " + name + ": ");
    int amount = input.nextInt();
    return amount;
}
```

2 input parameters: Scanner + String

```
public static void getAmount(Scanner input, String[] names, int[]
amounts) { // 2
    for (int i = 0; i < names.length; i++) {
        System.out.print("Enter the amount of " + names[i] + " : ");
        amounts[i] = input.nextInt();
    }
}
```

3 input parameters: Scanner + String pointer + int pointer

```
public static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    int sharp = getAmount(input, "Sharp");
    int brie = getAmount(input, "Brie");
    int swiss = getAmount(input, "Swiss");
    getAmount(input, names, amounts);
```

2 arguments: Scanner + String

3 arguments: Scanner + String[] + int[]

Type of arguments determines the method call!

Sum All

- Summation of numbers 1 to max
  - Steps
  - subTotal = 0;
  - subTotal += 1;
  - subTotal += 2;
  - ….
  - subTotal += max;
- Loop
  - Begin – 1
  - End – max
  - Increment – increase by 1
  - Body – add current number to running total

For-loop Forms

```
for (int i = 1; i <= max ; i++)
{
 subTotal += i;
}


for (int i = 0; i < max; i++) {
 subTotal += i + 1;
}


for (int i = max; i > 0; i--) {
 subTotal += i;
}
```

SumAll Method

1	2	3	4	5	6	7	8	9	10
1	3	6	10	15	21	28	36	45	55

```
public static int sumAll(int max) {
 int subTotal = 0;
 for (int i = 1; i <= max ; i++) {
  subTotal += i;
  System.out.println("sumAll "
  +  i + " value" + subTotal);
 }
 return subTotal;
}
```

in main ….

```
sumAll(5);
sumAll(10);
sumAll(20);
sumAll(15);
```

Run Result

```
sumAll 1 value 1
sumAll 2 value 3
sumAll 3 value 6
sumAll 4 value 10
sumAll 5 value 15
sumAll output for 5 is 15

sumAll 1 value 1
sumAll 2 value 3
sumAll 3 value 6
sumAll 4 value 10
sumAll 5 value 15
sumAll 6 value 21
sumAll 7 value 28
sumAll 8 value 36
sumAll 9 value 45
sumAll 10 value 55
sumAll output for 10 is 55
```

```
sumAll 1 value 1
sumAll 2 value 3
sumAll 3 value 6
sumAll 4 value 10
sumAll 5 value 15
sumAll 6 value 21
sumAll 7 value 28
sumAll 8 value 36
sumAll 9 value 45
sumAll 10 value 55
sumAll 11 value 66
sumAll 12 value 78
sumAll 13 value 91
sumAll 14 value 105
sumAll 15 value 120
sumAll 16 value 136
sumAll 17 value 153
sumAll 18 value 171
sumAll 19 value 190
sumAll 20 value 210
sumAll output for 20 is 210
```

```
sumAll 1 value 1
sumAll 2 value 3
sumAll 3 value 6
sumAll 4 value 10
sumAll 5 value 15
sumAll 6 value 21
sumAll 7 value 28
sumAll 8 value 36
sumAll 9 value 45
sumAll 10 value 55
sumAll 11 value 66
sumAll 12 value 78
sumAll 13 value 91
sumAll 14 value 105
sumAll 15 value 120
sumAll output for 15 is 120
```

Array of subTotals

```
public static int sumAll(int[] subTotal, int max) {

    for (int i = 1; i <= max ; i++) {
     if (subTotal[i] == 0) {//check if it contains the subTotal
       subTotal[i] = subTotal[i-1] + i;
       System.out.println("sumAll[" + i + "]
       value " + subTotal[i]);
     }
    }

    return subTotal[max];
}
```

Run Result

in main ….

```
sumAll(arr, 5);
sumAll(arr,10);
sumAll(arr, 20);
sumAll(arr, 15);
```

sumAll[1] value 1
sumAll[2] value 3
sumAll[3] value 6
sumAll[4] value 10
sumAll[5] value 15
sumAll output for 5 is 15
sumAll[6] value 21
sumAll[7] value 28
sumAll[8] value 36
sumAll[9] value 45
sumAll[10] value 55
sumAll output for 10 is 55
sumAll[11] value 66
sumAll[12] value 78
sumAll[13] value 91
sumAll[14] value 105
sumAll[15] value 120
sumAll[16] value 136
sumAll[17] value 153
sumAll[18] value 171
sumAll[19] value 190
sumAll[20] value 210
sumAll output for 20 is 210
sumAll output for 15 is 120

Arrays

- Require three steps to create and initialize:
  - Create pointer
    - int [] arr;
  - Create structure (array)
    - new int [SIZE];
    - it has to be pointed to by something
    - arr = new int [SIZE];
  - Can combine above two steps
    - int [] arr = new int [SIZE];
  - It contains SIZE entries with value 0
  - Third step to initialize to something else

```
for (int i = 0; i < arr.length; i++)
        arr[i] = somethingElse;
```

Class Variables (Lab #3)

- When a class variable is declared as ***private***, it can only be accessed within the class.

```
public class ScorerArr {

 private static int[] visitorScores;
 //Holds all the visitor scores in the array
 private static int[] homeScores;
 //Holds all the visitor scores in the array
 private static int inning;
 // the inning about to be played
 private static int batter;
 // the team about to bat (1 if visitors,2 if home team)
 …
}
```

Constants

- A constant variable is a variable declared as ***final***
  - Its value cannot be changed after being initialized.

```
public static final int REGULATION_NUM_INNINGS = 9;

 // assume game can never go over this number of innings
(used when allocating arrays)
 public static final int MAX_BOX_SCORE_LENGTH = 20;

 // value in a boxScore: an indicator that a half-inning
hasn't been played yet
 // this need to be distinguishable from a baseball score
 private static final int SENTINEL = -999;
```

SENTINEL	SENTINEL	SENTINEL	SENTINEL	SENTINEL	SENTINEL
SENTINEL	SENTINEL	SENTINEL	SENTINEL	SENTINEL	SENTINEL

For-loop (old type)

```
// Calculates the total score in the array teamBoxScore
// It ignores all the entries with the value SENTINEL
public static int gameScore(int[] teamBoxScore) {
    int output = 0;
    for (int i = 0; i < teamBoxScore.length; ++i) {
        if (teamBoxScore[i] != SENTINEL) {
            output += teamBoxScore[i];
        }
    }
    return output;
}
```

i is just used for indexing

For-loop (new type, called iterator)

```
// Calculates the total score in the array teamBoxScore
// It ignores all the entries with the value SENTINEL
public static int gameScore(int[] teamBoxScore) {
    int output = 0;
    for (int v : teamBoxScore) {
        // v is same as teamBoxScore[i]
        if (v != SENTINEL) {
        output += v;
        }
    }
    return output;
}
```

v is the value at each entry

of calls to gameScore

```
public static boolean gameIsOver ( ) {
    // Generic expression based on REGULATION_NUM_INNINGS so
    shorter games can be tested
    return (inning > REGULATION_NUM_INNINGS && batter == 1 &&
        gameScore(visitorScores) != gameScore(homeScores))

        || (inning == REGULATION_NUM_INNINGS && batter == 2 &&
        gameScore(visitorScores) < gameScore(homeScores));
}
```

What if this part is true?

Main

```
public static void main (String [ ] args) {

    // main has been filled in case your solution for
    lab 2 is overly complicated
    initialize ( );
    readScores (new Scanner (System.in));
    System.out.println (result ( ));
```