CSE 21
Intro to Computing II

Lecture 3 – Methods(2)

Announcement

- Lab 2 due before start of next lab
- Tutoring services
  - 10am – 12pm Thursdays
  - Location: TBD
- Reading assignment
  - Chapter 6.1 to 6.11 of textbook

Sum Example: Scope

```
public class SumExample{
  // Method Declaration like variables (callee)
  public static int sum(int num1, int num2) {                          #6
    System.out.println("Num1 is " + num1);                             #7
    System.out.println("Num2 is " + num2);                             #8
    int total = num1 + num2;                                           #9
    return total;                                                      #10
  }

  public static void main(String[] args) {                             #1
    int num1 = 18, num2 = 13;                                          #2
    System.out.println("Main num1 is " + num1);                        #3
    System.out.println("Main num2 is " + num2);                        #4
    int total #11 = sum(num2, num1);  //(caller switched arguments)    #5
     System.out.println("Sum is " + total);                            #12
  }
}
```

Output: Main num1 is 18
Main num2 is 13
Num1 is 13
Num2 is 18
Sum is 31

Variables : Scope

```
// Method Declaration like variables (callee)
public static int sum(int num1, int num2) {
  System.out.println("Num1 is " + num1);
  System.out.println("Num2 is " + num2);
  int total = num1 + num2;
  num1 = 100;
  return total;
}

public static void main(String[] args) {
  int num1 = 18, num2 = 13;
  int total = sum(num2, num1); // (caller)
  System.out.println("Main num1 is " + num1);
  System.out.println("Main num2 is " + num2);
  System.out.println("Sum is " + total);
  }
}
```

No Effect : Logical Error

Two sets of variables:
num1, ***num2*** and ***total*** local to each method are completely independent!

Multiple Returns (1)

```
public static int max(int num1, int num2) {                #3
    if (num1 > num2)                                       #4
        return num1;
    if (num2 > num1)                                       #5
        return num2;                                       #6
    if (num2 == num1)
        return num2;
}

public static void main(String[] args) {                   #1
    int maxNumber #7 = max(5, 10);                         #2
    System.out.println("Max is " + maxNumber);             #8
}
```

Multiple Returns (2)

```
public static int max(int num1, int num2) {                #3
    if (num1 > num2)                                       #4
        return num1;                                       #5
    if (num2 > num1)
        return num2;
    if (num2 == num1)
        return num2;
}

public static void main(String[] args) {                   #1
    int maxNumber #6 = max(15, 10);                        #2
    System.out.println("Max is " + maxNumber);             #7
}
```

Multiple Returns (3)

```
public static int max(int num1, int num2) {                #3
    if (num1 > num2)                                       #4
        return num1;
    if (num2 > num1)                                       #5
        return num2;
    if (num2 == num1)                                      #6
        return num2;                                       #7
}

public static void main(String[] args) {                   #1
    int maxNumber #8 = max(20, 20);                        #2
    System.out.println("Max is " + maxNumber);             #9
}
```

Multiple Returns Optimized (if)

```
public static int max(int num1, int num2) {                #3
    if (num1 > num2)                                       #4
        return num1;

    return num2;                                           #5
}

public static void main(String[] args) {                   #1
    int maxNumber #6 = max(5, 10);                         #2
    System.out.println("Max is " + maxNumber);             #7
}
```

Multiple Returns (Conditional)

```
public static int max(int num1, int num2) {                #3
    return num1 > num2 ? num1:num2;                        #4
}

public static void main(String[] args) {                   #1
    int maxNumber #5 = max(5, 10);                         #2
    System.out.println("Max is " + maxNumber);             #6
}
```

Name Overloading

- Name resolution is Scope dependent
- Variables just use the name
- Methods are declared and invoked using parentheses ( )
- Both require types

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!

Matching method calls

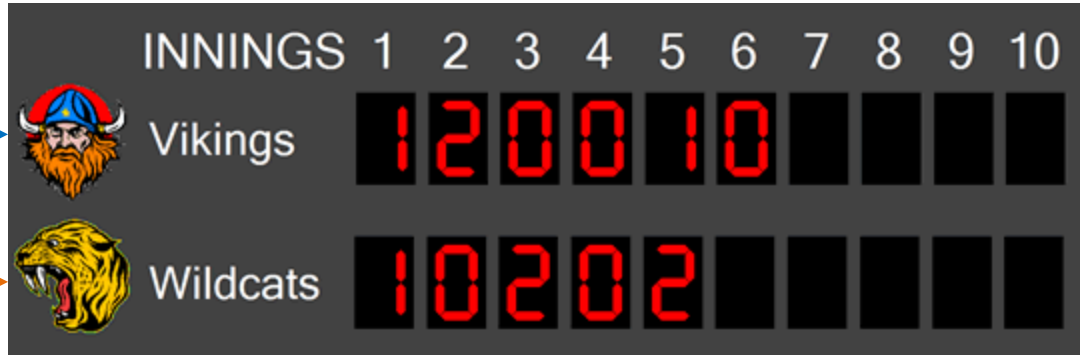
- getAmount(input, “Random”);
  - Scanner + String // Match 1
- getAmount(“Random”, input);
  - String + Scanner // Don’t match
- getAmount(input, names[0]);
  - Scanner + String // Match 1
- getAmount(input, names);
  - Scanner + String[] // Don’t match
- getAmount(input, names, amounts);
  - Scanner + String[] + int[] // Match 2
- getAmount(input, amounts, names);
  - Scanner + int[] + String[] // Don’t match
- getAmount(input,names[0],amounts[0]);
  - Scanner + String + int // Don’t match
- getAmount(input, names, new int[MAXCHEESE]);
  - Scanner + String[] + int[] // Match 2
- getAmount(input, names, prices);
  - Scanner + String[] + double[] // Don’t match

Lab #2

- Baseball scoring
  - Each game has 9 innings
    - First half of each inning: visitor's scores
    - Second half of each inning: home scores
  - Sum all inning scores for each team to determine who wins
  - In your program, scores are entered in one line:
    - 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 2
    - *The home team won 3 to 1 in 9 innings.*

Visitor →

Home →



Class Variables

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

```
public class Scorer {

 private static int visitorScore;

 private static int homeScore;

 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)
 …
}
```

Lab #2

- readScores method
  - It takes in a scanner input, which contain the sequence of scores.
  - While the game is not over:
    - Takes turn to add to `visitorScore` and `homeScore`
    - Needs to keep track on
      - Whose score you are reading (batter)
      - Current inning