CSE 21
Intro to Computing II

Lecture 5 – Methods (4)

Object Oriented Programming (1)

Today

- Methods (wrap up) and Object Oriented Programming
- Lab
  - Lab 5 due this week (2/23 – 3/3)
  - Midterm review labs this week
    - Attendance mandatory to receive full credit
  - Project 1 due next week, Friday March 9th at midnight
- Reading Assignment
  - Sections 7.1 – 7.8 (including participation activities)
    - Work on the **Participation Activities** in each section to receive participation grade at the end of semester (based on at least 80% completion)
    - Work on **Challenge Activities** to receive extra credit
  - Participation and Challenge activities evaluated at the end of semester

Midterm Details

- When and where
  - Date: Monday, March 05, 2018
  - Time: 3:30pm - 4:20pm
  - Location: CLSSRM 102
- Coverage
  - User-Defined Methods (6.1 - 6.11)
  - Lectures 1 - 5 (Excluding Object Oriented Programming)
  - Labs 1 - 6
  - Knowledge of all material from CSE 20 (assumed)
- Exam Policies
  - Open notes, and open book (chapter print-outs)
  - No electronic devices allowed
- Required material for exam: CatCard

Sum All (review)

- 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 (review)

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

i = 1, 2, 3, …, max (#iterations = max)

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

i = 0, 1, 2, …, max-1 (#iterations = max)

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

i = max, max-1, max-2, …, 1 (#iterations = max)

Be aware of how many iterations the loop runs!

SumAll Method

```
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

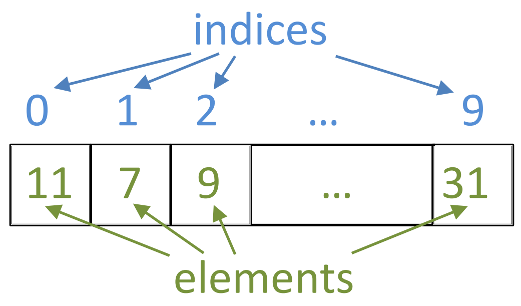
```
sumAll 1 value 1
sumAll 2 value 3
sumAll 3 value 6
sumAll 4 value 10
sumAll 5 value 15
sumAll 6 value 21
sumAll 7 value 28
sumAll 8 value 36
sumAll 9 value 45
sumAll 10 value 55
sumAll 11 value 66
sumAll 12 value 78
sumAll 13 value 91
sumAll 14 value 105
sumAll 15 value 120
sumAll 16 value 136
sumAll 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

Understanding Arrays

- One variable storing a list of data items

  ```
  int[] arr = {11, 7, 9, 4, 55, 2, 1, 18, 2, 31};
  ```

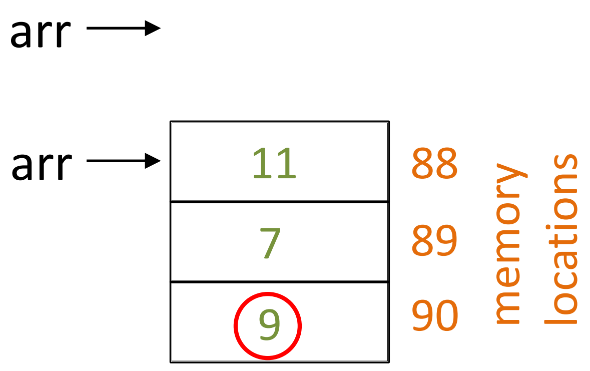


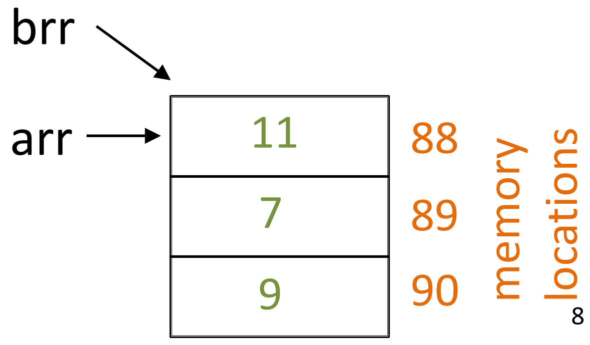
- Another view of arrays
  - An array variable is a *reference variable*
    - A pointer to a memory location

      ```
      int[] arr; //create pointer

      arr = new int[3]; // create structure
      arr[0] = 11; arr[1] = 7; arr[2] = 9;
      ```

    - Internally arr "stores" the memory location 88
    - When we write arr[2], internally we retrieve the element stored at memory location 88 + 2 (in this case, 9)
    - How about two variables pointing to the same array?

      ```
      int[] brr;
      brr = arr; // brr now "stores 88" as well
      ```





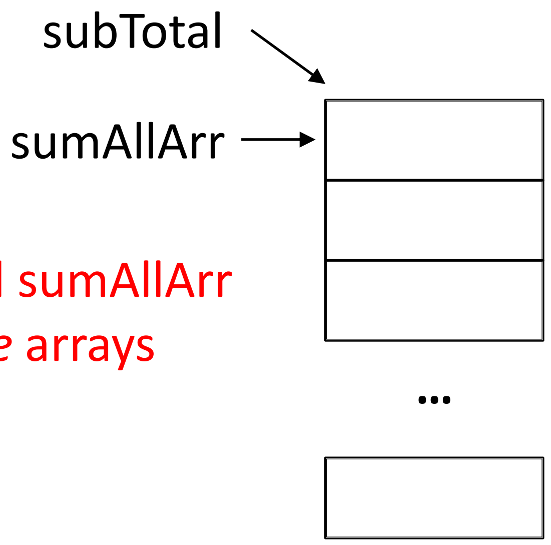
Array of subTotals

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

    for (int i = 1; i <= max ; i++) {
        if (subTotal[i] == 0) {              // Empty slot...calculate
            subTotal[i] = subTotal[i-1] + i;
            System.out.println("sumAll[" + i + "] value is " + subTotal[i]);
        }
    }

    return subTotal[max];
}
```

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

Array parameter in Methods

```
public static void sumAll(int[] subTotal, int max) {
        for (int i = 1; i <= max; i++)
                if(subTotal[i] == 0)
                        subTotal[i] = subTotal[i-1] + i;
}

public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int[] sumAllArr = new int[1000];
        for (int i = 0; i < 1000; i++) sumAllArr[i] = 0;
        int repeat = 0;
        do {
                System.out.print("Enter the max number for sumAll (between 0 and 1000): ");
                int max = input.nextInt();
                sumAll(sumAllArr, max);
                for (int i = 0; i <= max; i++)
                        System.out.println("sumAllArr[" + i + "] value is " + sumAllArr[i]);
                System.out.print("Repeat this program? (1 for yes) ");
                repeat = input.nextInt();
        } while (repeat == 1);
}
```



subTotal and sumAllArr are the *same* arrays

Run Result

In main() ….

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

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

For-loop (old vs new)

```
// Calculates the total score in the array teamBoxScore
// It ignores all the entries with the value SENTINEL (initialized as final elsewhere)
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

```
public static int gameScoreV2(int[] teamBoxScore) {
        int output = 0;
        for (int bscore : teamBoxScore) { // bscore is same as teamBoxScore[i]
                if (bscore != SENTINEL) {
                        output += bscore;
                }
        }
        return output;
}
```

bscore is the element at each array index

How we did things so far...

- One program with a single main() method doing everything
- As programs got bigger, we started building them up using a ***composition*** of methods
  - The starting point is still main()
  - Methods do some of the work, and then combine everything in main()
  - We've been doing this with static methods
- We'll now see a completely new way of thinking about the programming world!
  - It is a complete paradigm shift

Object-Oriented Programming

- Our new programming metaphor is multiple independent intelligent agents called ***Objects***
- An object can…
  - ask other objects to do things
    - this is called "message passing"
  - remember things about its own past history
    - this is called "local state"
  - behave just like another except for a few differences
    - this is called "inheritance"
- Many people find this way of thinking and modeling the world more intuitive
  - The world is made up of objects! people, desks, chairs, etc.
- Android Apps are built this way!

What is an Object?

- Real-world objects share states and behaviors
  - E.g., cats have states (name, color, breed, hungry) and behaviors (meowing, sleeping, shredding rugs)
  - E.g., bikes have states (gear, # wheels, # gears) and behaviors (braking, changing gears)
- Software objects are modeled after real-world.
- A software object…
  - maintains its states in one or more variables
  - implements its behaviors with methods
  - An object is a software bundle of variables (what it knows) and related methods (what it can do)
- ***Classes*** are **"factories"** for generating objects