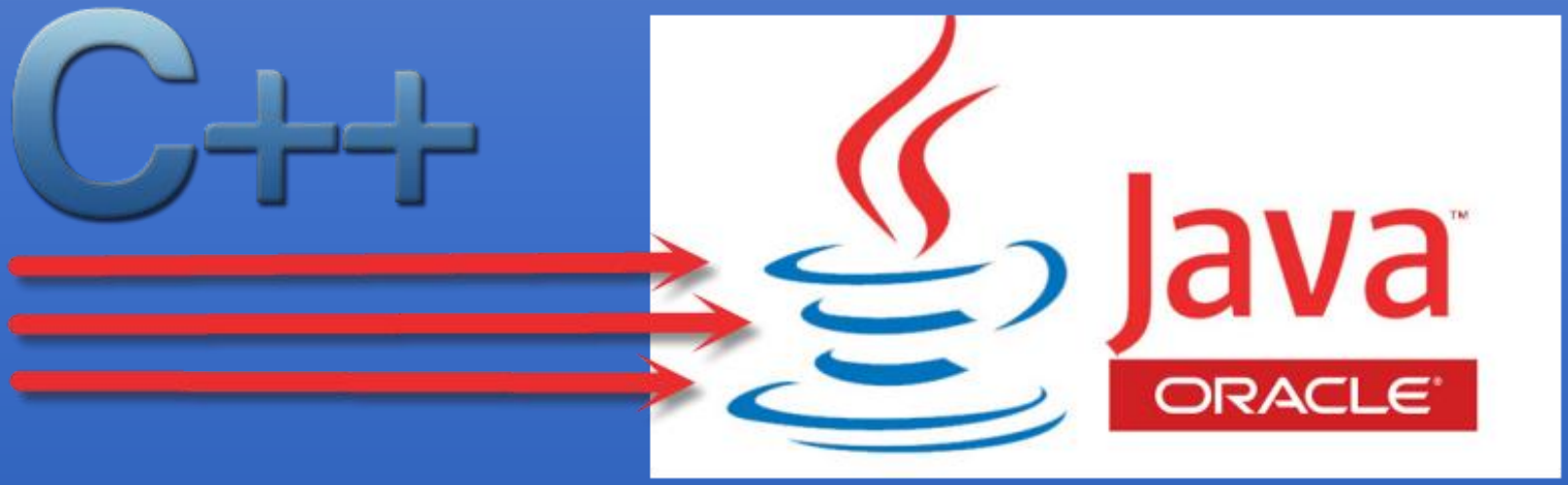


Lab 2:
Java Basic for C++ Programmer

Basic lab instructions

- Talk to your classmates for help. You can even work on the lab with a partner if you like.
- You may want to bring your textbook to future labs to look up syntax and examples.
- Stuck? Confused? Have a question? Ask a LA for help, or look at the book or past lecture slides.
- Complete as many problems as you can within the allotted time. You don't need to keep working on these exercises after you leave the lab.
- Feel free to complete problems in any order.
- Tell me when you finish the exercises.

Goals

- ✓Use java basic Data types
- ✓Use java basic Operators
- ✓Trace and write **for loops** for repeating lines of code
- ✓Practice using for loops to traverse and process array data

I. KEYNOTE

Java Basic Data Types

- Primitive Data Types:
  - byte: a 8-bit signed integer.
  - short: a 16-bit signed integer.
  - int: a 32-bit signed integer.
  - long: a 64-bit signed integer.
  - float: a single-precision 32-bit floating point.
  - double: a double-precision 64-bit IEEE 754 floating point.
  - boolean: one bit of information.
  - char: a single 16-bit Unicode character.
- Reference Data Types:
  - Reference variables are created using defined constructors of the classes. **Class objects**, and various type of array variables come under reference data type.
  - **Default value** of any reference variable is **null**.
  - A **reference variable** can be used to refer to any object of the declared type or any compatible type.
  - Example: Animal animal = new Animal("giraffe");

Java Basic Operators

- The Arithmetic Operators:

  `+, -, *, /, %, ++, --`

  - The Relational Operators:

    `==, !=, >, <, >=, <=`

  - The Bitwise Operators:

    `&(AND), |(OR), ^(XOR), ~(Bit flipping), <<(Bit shift left), >>(Bit shift right), >>>(Shift right zero fill)`

  - The Logical Operators:

    `&&(AND), ||(OR), !(NOT)`

  - The Assignment Operators:

    `=, +=, -=, *=, /=, %=, <<=, >>=, &=, ^=, |=`

  - Others:
    - Conditional Operator (?:)
      i.e. x=(condition) ? true : false
    - instanceof() Operator
      i.e. when String name = "James" then name instanceof(String) return true.

Java Basic Operators (example)


```java
public class Test {
   public static void main(String args[]) {
        int a, b;
        a = 10;
        b = (a == 1) ? 20 : 30;
        System.out.println("Value of b is : " + b);
        // Value of b is : 30
        b = (a == 10) ? 20 : 30;
        System.out.println("Value of b is : " + b);
        // Value of b is : 20
   }
}
```

```java
class Vehicle {}

public class Car extends Vehicle {
   public static void main(String args[]) {
        Vehicle a = new Car();
        boolean result = a instanceof Car;
        System.out.println(result); // return true
   }
}
```

Java Basic Operators (precedence)

- Precedence of Java Operators:

Category	Operator	Associativity
Postfix	() [] . (dot operator)	Left to right
Unary	++ - - ! ~	Right to left
Multiplicative	* / %	Left to right
Additive	+ -	Left to right
Shift	>> >>> <<	Left to right
Relational	> >= < <=	Left to right
Equality	== !=	Left to right
Bitwise AND	&	Left to right
Bitwise XOR	^	Left to right
Bitwise OR	|	Left to right
Logical AND	&&	Left to right
Logical OR	||	Left to right
Conditional	?:	Right to left
Assignment	= += -= *= /= %= >>= <<= &= ^= |=	Right to left
Comma	,	Left to right

Java Types of Loops


```
for(initialization; Boolean_expression; update){
   //Statements
}
```

Execute a specific number of times.

```
for(declaration : expression){
   //Statements
}

String[] namelist = { “Kim", “Park", “Lee"};
for (String name : namelist) {
   System.out.print(name);
   System.out.print(",");
}
```

```
while(Boolean_expression){
   //Statements
}
```

Guarante to execute at least one time.

```
do{
   //Statements
}while(Boolean_expression);
```

Java Array

- Declaring Array Variables

  `dataType[] arrayRefVar;` // preferred way.
  or
  `dataType arrayRefVar[];` // works but not preferred way.

- Creating Arrays
  - `arrayRefVar = new dataType[arraySize];`
  - `dataType[] arrayRefVar = new dataType[arraySize];`
- Alternatively you can create arrays as follows:
  - `dataType[] arrayRefVar = {value0, value1, ..., valuek};`
- Passing Arrays to Methods:
  - `public static type name(type[] name)`
- Returning an Array from a Method:
  - `public static type[] name(parameters)`

Java Array (methods)

- The java.util.Arrays class contains various static methods for sorting and searching arrays, comparing arrays, and filling array elements. These methods are overloaded for all primitive types.

Method name	Description
Arrays.binarySearch(array, value)	returns index of value in a sorted array (< 0 if not found)
Arrays.copyOf(array, length)	returns a new copy of an array
Arrays.equals(array1, array2)	returns true if the two arrays contain same elements
Arrays.fill(array, value)	sets every element to the given value
Arrays.sort(array)	arranges the elements into sorted order
Arrays.toString(array)	returns a string for the array, such as "[10, 30, -25, 17]"

Java Array (example)


```
public class ArrayTutorial
{
    public static void main(String[] args)
    {
        // METHOD 1: All the elements of this array are initialized to 0 by default
        int[] array = new int[5];

        // METHOD 2: This way provides initial values for the arrays
        int[] array2 = {1,3,6,4,5};

        // METHOD 3: Similar to method 1, but some people prefer this
        final int SIZE = 5;
        int[] array3 = new int[SIZE];

        // print the array value
        int   cnt = 0;
        for (cnt = 0; cnt < array2.length; cnt++) {
            // With the cnt variable I am now indexing through the array
            System.out.print(array2[cnt] + "\n");
        }

        // sort a list
        Arrays.sort(array2);
        // print a string representation of an array
        System.out.println(Arrays.toString(array2));
    }
}
```

Java Random class


```
// import the library
import java.util.Random;


/** Generate 10 random integers in the range 0..99. */

public final class RandomInteger {


    public static void main(String[] args) {

        System.out.print("Generating 10 random integers in range 0..99.");


        // create a single Random object which is reused in here

        Random randomGenerator = new Random();

        for (int idx = 1; idx <= 10; ++idx) {

            int randomInt = randomGenerator.nextInt(100);
            // randomGenerator.nextLong();

            // randomGenerator.nextFloat();

            // randomGenerator.nextDouble();

            // randomGenerator.nextGaussian();

            System.out.print("Generated : " + randomInt);

        }

    }

}
```

II. PRACTICE

Exercise 1
Multiplication Quiz App

- Objective:
  - Declaring and using methods.
  - Using class Random objects.
  - Using switch statement.
- Problems:
  - Write a program that will help an elementary school student learn multiplication. Use a Random object to produce two positive one-digit integers. The program should then prompt the user with a question, such as
    - How much is 6 times 7?
  - The student then inputs the answer. Next, the program checks the student's answer.
    - If it is correct, display the message "Very good!" or "Excellent!" or "Nice work!", and ask another multiplication question.
    - If the answer is wrong, display the message "No. Please try again." or "No. Keep trying." or "Don't give up!", and let the student try the same question repeatedly until the student finally gets it right.
- Note
  - It's better for student to create methods `createQuestion()` and `createResponse()` when coding.

Exercise 1
Multiplication Quiz App (cont)

- Sample output:

```
How much is 6 times 7?
Enter your answer (-1 to exit):37
No. Keep trying.
Enter your answer (-1 to exit):42
Nice work!
How much is 8 times 3?
Enter your answer (-1 to exit):-1
```

- Extend your app
  - Add other operations (+, - , <<, >> …)
  - Make multiple choices answer.
  - Make test app and give a grade (A, B, C, D, F) to a tester at the end of the app.

Exercise 2
Roll the dices

- Objective:
  - Using class Random objects.
  - Using array to store set of data.
- Problem:
  - Write an application to simulate the rolling of n dices. The application should use an object of class Random to roll k times. The sum of the two values should then be calculated.
  - Each die can show an integer value from 1 to 6, so the sum of the values will vary from n to 6n.
  - Display the results in tabular format.
- Note:
  - Number of dice n and number of roll k is input by user.
  - Use a one dimensional array to keep track of the number of times each possible sum appears.

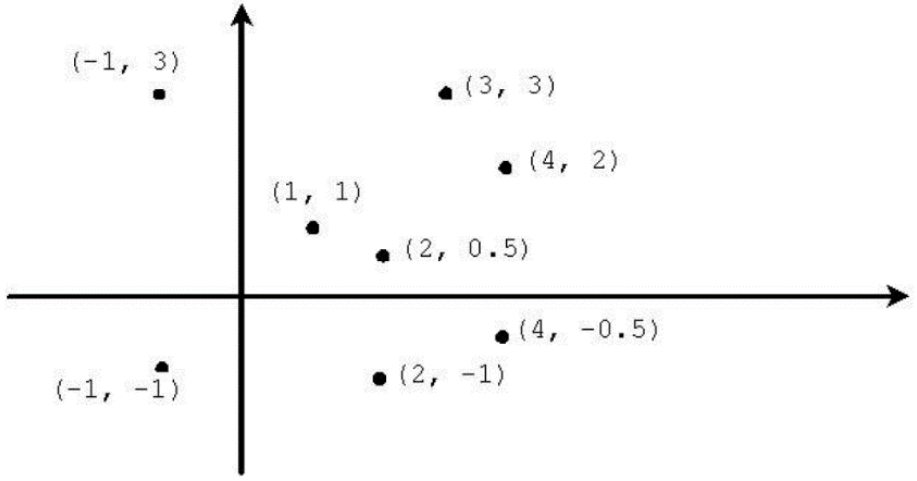
Exercise 2
Roll the dices (cont)

- Sample output:

```
Enter number of dices: 3
Enter number of rolls: 1000000
Sum    Frequency  Percentage
  3         4642        0.46
  4        13883        1.39 *
  5        27600        2.76 ***
  6        46141        4.61 ****
  7        69426        6.94 *******
  8        97113        9.71 **********
  9       115567       11.56 ***********
 10       125452       12.55 ************
 11       125433       12.54 ************
 12       115513       11.55 ***********
 13        96779        9.68 **********
 14        69469        6.95 *******
 15        46494        4.65 *****
 16        27924        2.79 ***
 17        13924        1.39 *
 18         4640        0.46
```

Exercise 3
Finding a Closest Pair

- Objective:
  - Using class Multidimensional Array
- Problem:
  - Write an application to input the number of points and then points' positions (x and y).
  - Calculate and display the closest pair of point.



	x	y
0	-1	3
1	-1	-1
2	1	1
3	2	0.5
4	2	-1
5	3	3
6	4	2
7	4	-0.5

Exercise 3
Finding a Closest Pair

- Sample output:

```
Enter the number of points: 8
Enter 8 points: -1 3 -1 -1 1 1 2 0.5 2 -1 3 3 4 2 4 -
0.5
The closest two points are (1.0, 1.0) and (2.0, 0.5)
with the distance is 1.118
```