CSE2101: Object Oriented Programming-II (Java)

Lecture 3

Operators, Array and String

Operators

Java Operators

- 4 groups:
  - Arithmetic
  - Bitwise
  - Relational
  - logical

Arithmetic Operators

- Operators: +, -, *, /, %, ++, --, +=, -=, *=, /=, %=
- For integers:
  - Division is integer division
    - 6 / 2 yields 3
    - 7 / 2 yields 3, not 3.5
  - Modulus is %
    - Returns the remainder
    - 7 % 2 yields 1
    - 6 % 2 yields 0
- For Floats and doubles
  - Division
    - 7.0 / 2.0 yields 3.5
    - 7.0 / 2 yields 3.5
    - 7 / 2.0 yields 3.5
    - 7 / 2 yields 3

- Modulus:
  - Differs from C/C++
  - 47.5 % 10 yields 7.5

+=, ++

- += is more efficient than +
  - A=A+5;   A+=5;
- ++
  - Increments a number variable by 1
- --
  - Decrements a numeric variable by 1
- Output:

```
int i = 4,j,k;
j=++i; //prefix form
k=i++; //postfix form
```

System.out.println

- System.out.println (“result: “ + 3/5);
  - What does it print?
  - result: 0
- System.out.println (“result: “ + 5 % 3);
  - What does it print?
  - result: 2
- System.out.println (“result: “ + 3/5.0);
  - What does it print?
  - result: 0.6
- System.out.println (“result: “ + 3+4.0);
  - What does it print?
  - result: 34.0
- System.out.println (“result: “ + (3+4.0));
  - What does it print?
  - result: 7.0

Bitwise operators

- Bitwise operator
  - | - or
  - & - and
  - ~ - Not
  - ^ - XOR
  - >> - shift right
  - << - shift left
  - >>> - shift right zero fill

Example of Bitwise and Relational Operator

		Results
• byte a=8, b=24, c ;		
• c=a | b;	$\longrightarrow$	00001000 | 00011000 = 24
• c=a & b;	$\longrightarrow$	00001000 & 00011000 = 8
• c=~a;	$\longrightarrow$	~00001000=11110111=-119
• c=a^b;	$\longrightarrow$	00001000 ^ 00011000 = 16
• c=a<<1;	$\longrightarrow$	00001000 << 1= 16
• c=a>>2;	$\longrightarrow$	00001000 >> 2= 2
• c=a>>>1;	$\longrightarrow$	00001000 >>> 1= 4

Relational Operator

- Relational operators:==, !=, >, <, >=, <=
- The outcome of these operations is a boolean value. So the outcome is not numeric value.
- true and false are non-numeric values.
- Example 1:

```
int a=4, b=1;
boolean c= a<b;            //The result of c will be false
```

- Example 2:

```
int done = 3;
if(done)                   //error
    System.out.println("abc");
if(done!=0)                //ok
    System.out.println("abc");
```

Defining boolean variables

- Local boolean variables with initialization

```
boolean canProceed = true;
boolean preferCyan = false;
boolean completedSecretMission = true;
```

canProceed	true
preferCya n	false
completedSecretMission	true

Boolean Logical Operators

- Works on boolean values.
- Operators: |, &, ^, ||, &&, !, ?:, !=, ==,
- Suppose

```
boolean p = true;
boolean q = false;
boolean r = true;
boolean s = false;
```

- What is the value of

```
p                 p && s
!s                p == q
q                 q != r
p && r                 r == s
q || s                 q != s
```

Evaluating boolean expressions

- Suppose

```
int i = 1;
int j = 2;
int k = 2;
char c = '#';
char d = '%';
char e = '#';
```

- What is the value of

```
j == k          i != k
i == j          j != k
c == e          d != e
c == d          c != e
```

Short-circuit Logical Operators

- || and && are the short-circuit operators.
  - Java will not evaluate the right-hand operand when the outcome of the expression can be determined by the left operand alone.
- | and & are not the short-circuit operators.
  - Java evaluates both sides of the operator.
- Example:
  - if(a!=0 && b/a > 10) {...}
    - If a==0, right part will not be evaluated.
  - if(a!=0 & b/a > 10) {...}
    - If a==0, the program returns run-time exception.
- We should use || and &&

Assignment vs. comparison

- = is the assignment operator
- == is the comparison operator
  - Returns a boolean (true or false) if the two sides are equal
  - Consider:

```
int x = 5;
System.out.println (x == 5);
System.out.println (x == 6);
```

  - Prints out true, false

Operator precedence revisited

- Highest to lowest
  - Parentheses
  - Unary operators
  - Multiplicative operators
  - Additive operators
  - Relational ordering
  - Relational equality
  - Logical and
  - Logical or
  - Assignment

Taking Input

Taking input from the keyboard

- Here Scanner class is used to take input from the keyboard.
- Scanner is a simple text scanner which can parse primitive types and strings using regular expressions.
- First, Scanner class is connected to System.in
- Then, it uses it's internal functions to read from System.in
- Scanner class is under the package of java.lang.util
- Example:

```
Scanner sc = new Scanner(System.in);
int i;
If(sc.hasNextInt()==true)
        i = sc.nextInt();
else{
}
```

Take an input from the keyboard

```
import java.util.*;
public static void main(String[] args) {
       double value;
  System.out.print(“Enter a floating point number:");
       Scanner stdin = new Scanner(System.in);
       if(stdin.hasNextDouble()==true)
              value=stdin.nextDouble();
       System.out.println(“You have entered: ”+value);
}
```

Scanner API

```
public Scanner(InputStream in)      // Scanner(): convenience constructor for an
                                    // InputStream

public Scanner(File s)              // Scanner(): convenience constructor for a filename

public int nextInt()                // nextInt(): next input value as an int

public short nextShort()            // nextShort(): next input value as a short

public long nextLong()              // nextLong(): next input value as a long

public double nextDouble()          // nextDouble(): next next input value as a double

public float nextFloat()            // nextFloat(): next next input value as a float

public String next()                // next(): get next whitespace-free string

public String nextLine()            // nextLine(): return contents of input line buffer

public boolean hasNext()            // hasNext(): is there a value to next
```

Another Example

```
import java.util.*;

public class MathFun {

        public static void main(String[] args) {
                // set up the Scanner object
                Scanner stdin = new Scanner(System.in);

                // have the user input the values for x and y
                System.out.print("Enter a decimal number: ");
                double x = stdin.nextDouble();
                System.out.print("Enter another decimal number: ");
                double y = stdin.nextDouble();

                double squareRootX = Math.sqrt(x);

                System.out.println ("Square root of " + x + " is "
                                        + squareRootX);
        }
}
```

Arrays

Background

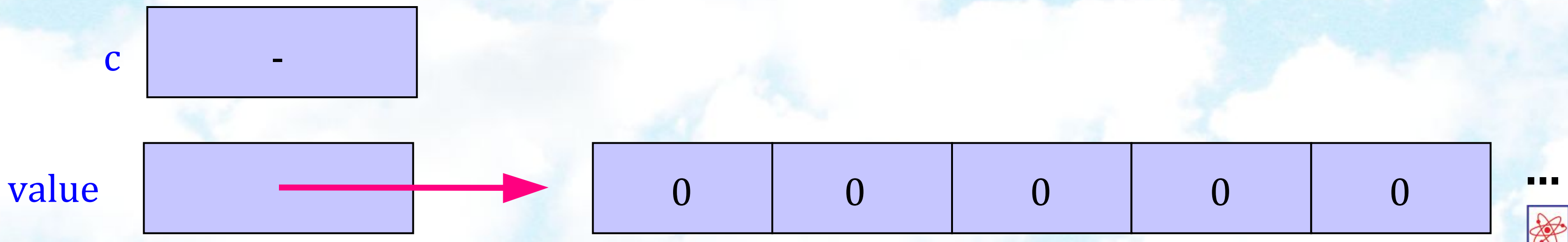
- Programmer often need the ability to represent a group of values as a list
  - List may be one-dimensional or multidimensional
- Java provides arrays and the collection classes
  - The Vector class is an example of a collection class
- Consider arrays first

Example

- Definitions

```
char[] c;
int[] value = new int[10];
```

- Causes
  - Array object variable c is un-initialized
  - Array object variable value references a new ten element list of integers
    - Each of the integers is default initialized to 0



An array example

```
int[] v = new int[10];
int i = 7;
int j = 2;
int k = 4;
v[0] = 1;
v[i] = 5;
v[j] = v[i] + 3;
v[j+1] = v[i] + v[0];
v[v[j]] = 12;
System.out.println(v[2]);
v[k] = stdin.nextInt();
```

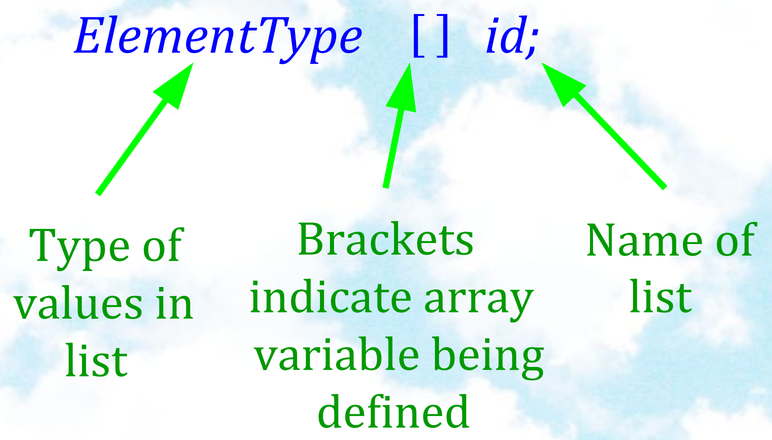
8 is displayed

Suppose 3 is extracted

v	1	0	8	6	3	0	0	5	12	0
	v[0]	v[1]	v[2]	v[3]	v[4]	v[5]	v[6]	v[7]	v[8]	v[9]

Array variable definition styles

- Without initialization



```
int [] a;
int a[];
```

Array variable definition styles

- With initialization

Nonnegative integer expression specifying the number of elements in the array

$$\textit{ElementType}\ [\ ]\ \ \textit{id}\ = \underbrace{\textbf{new}\ \ \textit{ElementType}\ [\overset{\downarrow}{n}];}_{\text{Reference to a new array of n elements}}$$

Where we've seen arrays

- public static void main (String[] args)
  - Thus, the main() method takes in a String array as the parameter

- Note that you can also define it as:
- public static void main (String args[])

Java array features

- Subscripts are denoted as expressions within brackets: [ ]
- Base (element) type can be any type
- Size of array can be specified at run time
  - This is different that pure C! (for the most part, at least)
- Index type is integer and the index range must be 0 ... n-1
  - Where n is the number of elements
- Automatic bounds checking
  - Ensures any reference to an array element is valid
- Data field length specifies the number of elements in the list
- Array is an object
  - Has features common to all other objects
  - More on this later…

Consider

- Segment

```
int[] b = new int[100];
b[-1] = 0;
b[100] = 0;
```

- Causes
  - Array variable to reference a new list of 100 integers
    - Each element is initialized to 0
  - Two exceptions to be thrown
    - -1 is not a valid index – too small
    - 100 is not a valid index – too large

      – IndexOutOfBoundsException

Explicit initialization

- Syntax

id references an array of n elements. id[0] has value exp_0, id[1] has value exp_1, and so on.

$$\mathit{ElementType}\ [\,]\ \ \mathit{id} = \{\ \underbrace{\mathit{exp}_{0}\ ,\ \mathit{exp}_{1}\ ,\ \ldots\ \mathit{exp}_{n\text{-}1}}\ \}\ ;$$

Each exp_i is an expression that evaluates to type ElementType

Explicit initialization

- Example

```
String[] puppy = { “pika“, “arlo“, “schuyler", “nikki" };
int[] unit = { 1 };
```

- Equivalent to

```
String[] puppy = new String[4];
puppy[0] = “pika";       puppy[1] = “arlo";
puppy[2] = “schuyler";   puppy[3] = “nikki";

int[] unit = new int[1];
unit[0] = 1;
```

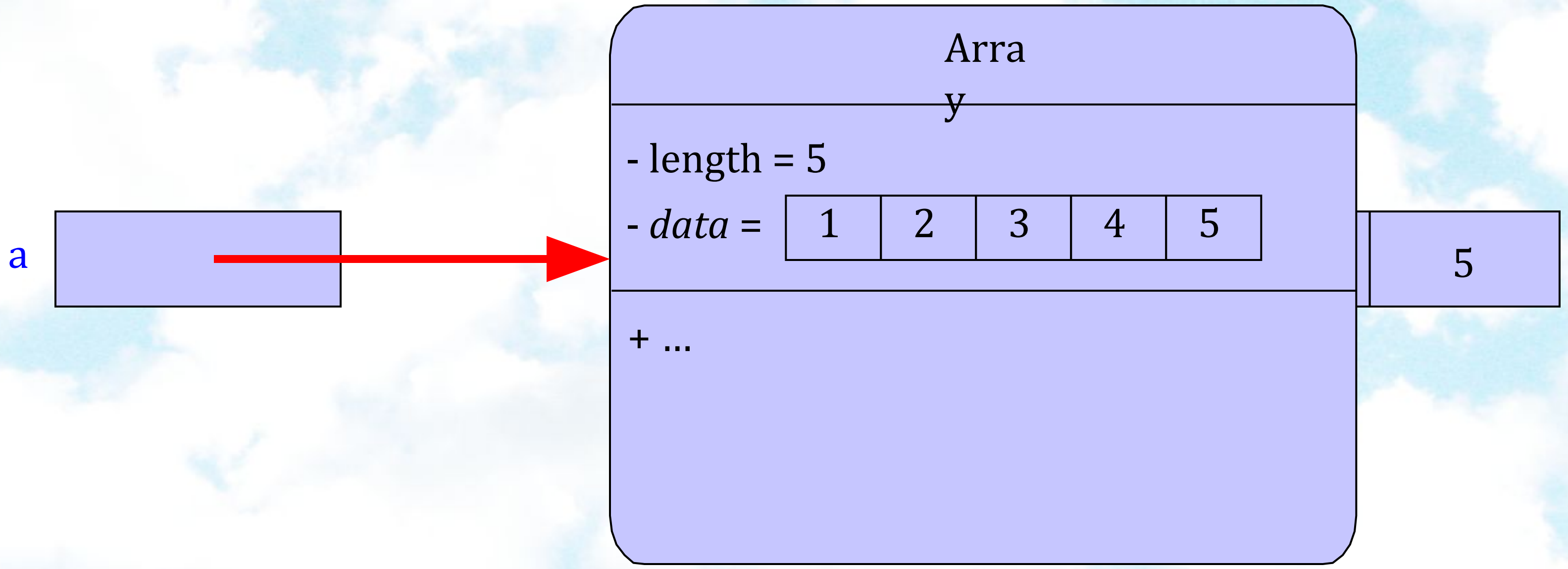
Review of arrays

- Creating an array:
```
int[] foo = new int[10];
```
- Accessing an array:
```
foo[3] = 7;
System.out.print (foo[1]);
```
- Creating an array:
```
String[] bar = new String[10];
```
- Accessing an array:
```
bar[3] = “qux”;
System.out.println (bar[1]);
```

How Java represents arrays

- Consider

```
int[] a = { 1, 2, 3, 4, 5 };
```

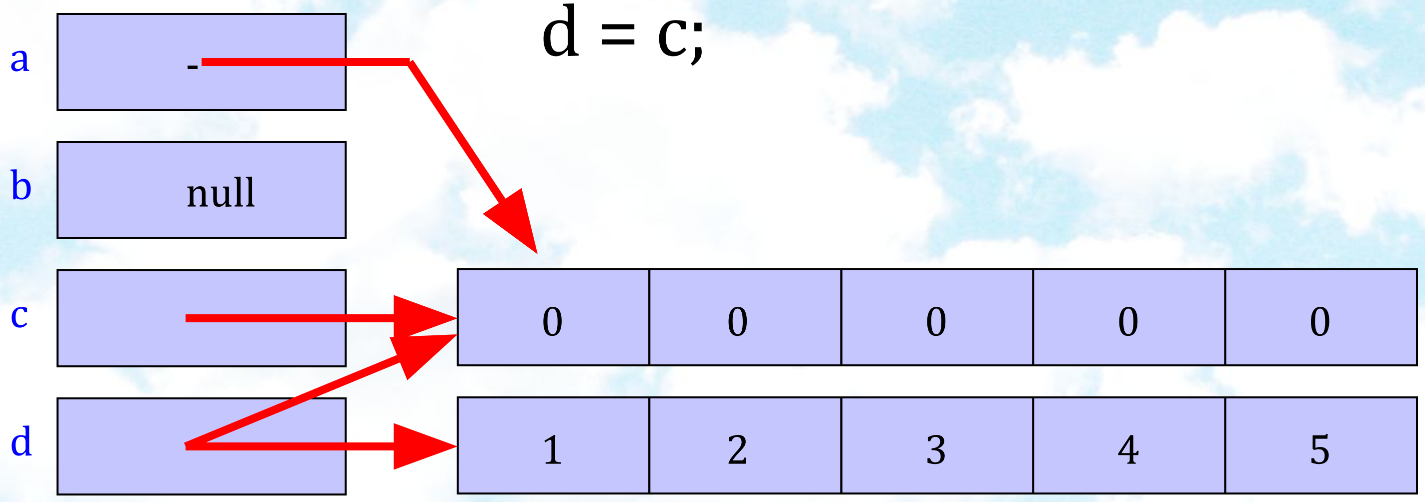


More about how Java represents Arrays

- Consider

```
int[] a;
int[] b = null;
int[] c = new int[5];
int[] d = { 1, 2, 3, 4, 5 };
a = c;
d = c;
```

```
int[] a;
int[] b = null;
int[] c = new int[5];
int[] d = { 1, 2, 3, 4, 5 };
a = c;
d = c;
```



Character Array Vs. String Class

Character Array

- Example:
- char myarray[]=new char[20];
- myarray[0]='a';
- myarray[1]='b';
- myarray[2]='\0';
- System.out.println(myarray);

Strings

- Java provides a class definition for a type called String
- Since the String class is part of the java.lang package, no special imports are required to use it (like a header file in C).
- Just like regular datatypes (and like C), variables of type String are declared as:
  - String s1;
  - String s2, s3; //etc.
- Note that String is uppercase. This is the Java convention for classnames.

Strings

- Initializing a String is painless
  - s1 = “This is some java String”;
- Note that double quotes are required.
- Memory is allocated dynamically.
- Think of above method as shortcut for more standard way (assuming s1 has been declared):
  - s1 = new String(“This is some java String”);
  - *new* operator required to create memory for new String object.

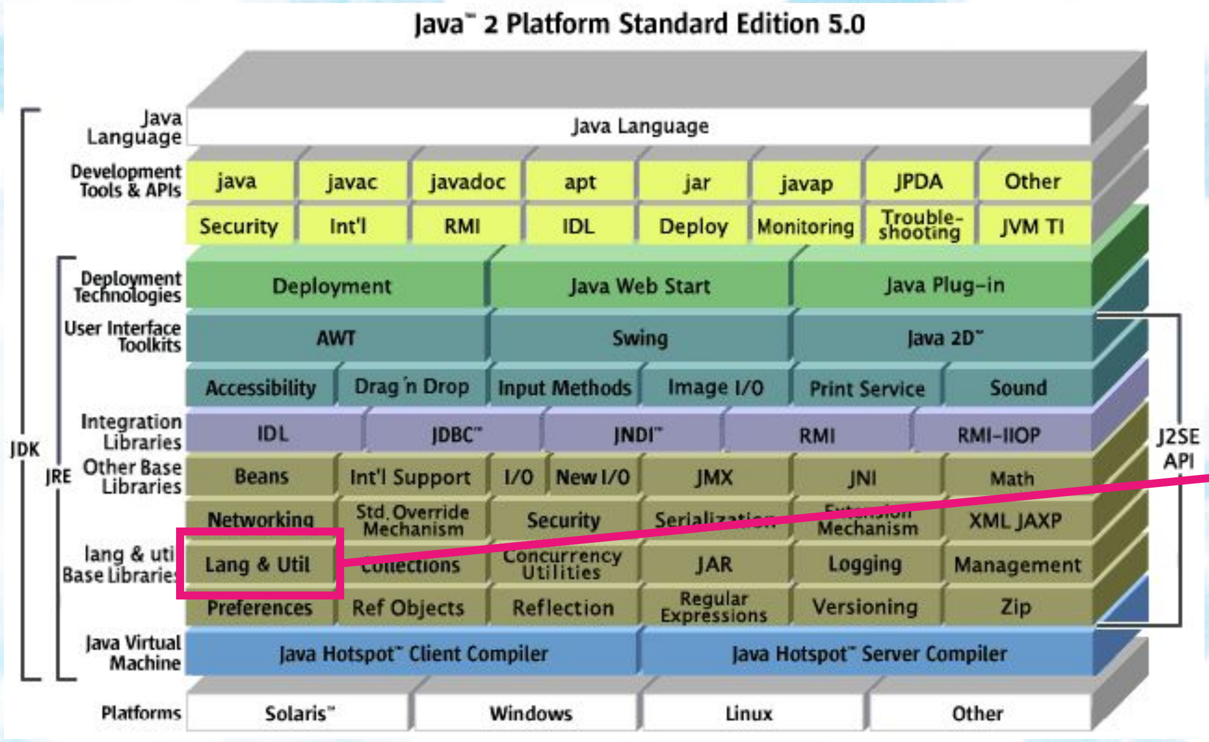
String Examples

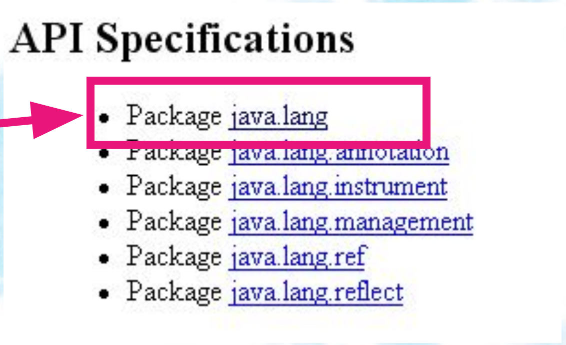
- Best to see by way of example:

```
String s = new String("Hello");
Char c = s.charAt(3);
System.out.println(c);
```

- Method charAt called on String object s taking single integer parameter.

How to get help (for String Class)





API Specifications

- Package java.lang
- Package java.lang.annotation
- Package java.lang.instrument
- Package java.lang.management
- Package java.lang.ref
- Package java.lang.reflect

Reading:

- Java2: The Complete Reference (Herbert Schildt)
  - Chapter 3: Data types, Variable and Arrays
  - Chapter 4: Operators
  - Chapter 13: String Handling

Homework (String and Scanner)

- Write a program that takes two string S1 and S2 as input and perform the following operations:
  - Print the length of each string.
  - Replace all spaces of S1 to underscore(_).
  - Print the first character of S1.
  - Compare the string S1 and S2 and print “equal” or “not equal” accordingly
  - Find the first occurrence of character ‘a’ in S1 and print it’s position.
  - If S1 is a substring of S2 or S2 is a substring of S1 then print a message.
  - Convert the S1 string to lower case and S2 string to upper case letter.
  - Save the S1 string to a character array.
- What is the task of “trim” function?
- What’s the difference between “equals()” and “==“ to compare string?

Thank you