COMP-248
Object Oriented Programming I

Arrays of/as Objects

Parts of the slides are taken from Prof. L. Kosseim

Next:

1. **Arrays are objects**
2. Arrays **of** objects

Arrays

- Two ways to view an array:
  - A **collection of indexed variables**
  - A **single item** whose value is a **collection of values** (of some base type)

- The **elements** of an array can be:
  - A primitive type or
  - An object reference (e.g., Date, Car)

- In Java, **the array itself is an object**
  - The name of the array is a reference
  - The array must be created with **new**

Arrays as Objects

- The syntax for creating an array object is a bit different, but very similar to declaring a new object

```
double [] a = new double[10];
```

- The expression **new double[10]** can be thought of as a **constructor** that uses a nonstandard syntax

- Every array has an instance variable called `length`

- Indexed variables are not instance variables, they are a 'special kind' of variable

Arrays are Objects

- Since an array is an object, it can be passed as a parameter to a method

- Like any other object:
  - The reference to the array is passed
  - Changing an array element within the method changes the original object

- Note: passing the entire array ≠ passing a single element of the array

Array Parameters

- Array **indexed variables** and **entire arrays** can be used as parameters

  - For example:

```
double n = 0;
double [] a = new double[10];
```

- If myMethod takes in as argument a single double, then
  - myMethod(n) is legal
  - myMethod(a[3]) is also legal

Example

```
public boolean isVowel(char someChar) {

if (someChar =='a' || someChar =='e' || …)
     {
          return true;
     }
          else
          return false;
}
```

```
public int nbVowel(char[] someCharArray) {

int nb = 0;
for (int i =0; i < someCharArray.length; i++)
     {
          if (isVowel(someCharArray[i]))
               nb++;
     }
     return nb;

}
```

```
char[] alphabet = {'a', 'b', 'c', 'd, …, 'z'};
if (isVowel(alphabet[0]))  // 1. OK ?
    System.out.print("…");
if (isVowel(alphabet))    // 2. OK ?
    System.out.print("…");

if (nbVowel(alphabet)>3)  // 3. OK ?
    System.out.print("…");

if (nbVowel(alphabet[4])>3) // 4. OK ?
    System.out.print("…");
```

Example

write a method to return the sum of an array of integers.

```
public static int arraySum (int[] y)
 {
  int sum =0;
  for (int i =0; i < y.length; i++)
      {
           sum += y[i];
      }
  return sum;
 }
```

Arrays and Assignment

- We cannot change the size of an array but we can assign an array to another…

```
int[] a1 = {10, 20, 30};
int[] a2 = {1, 2, 3, 4, 5};

System.out.println(a1.length);
System.out.println(a1[0]);
a1 = a2;
System.out.println(a1.length);
System.out.println(a1[0]);
```

```
3
10
5
1
```

Output

Duplicating/Copying an Array

To copy the content of an array into another:

```
static int[] duplicate(int[] theOriginal) {
  int[] theCopy = new int[theOriginal.length];
  for (int i = 0; i < theOriginal.length; i++)
    theCopy[i] = theOriginal[i];
  return theCopy;
}
```

Or use the built-in **`clone`** method (for 1-d arrays)

```
theCopy = (int[]) theOriginal.clone();
```

Note: **`clone`** returns an array to a generic **`Object`**, we need to cast the result to what we want (ex. **`int[]`**)

Example

```
public static void main(String[] args) {
    int[] a1 = {10, 20, 30};
    int[] a2 = {1, 2, 3, 4, 5};

    System.out.println(a1[0]);
     // a1 = duplicate(a2);
    // a1 = (int[]) a2.clone();
    // a1 = a2;
    System.out.println(a1[0]);
    a1[0] = 99;
    System.out.println(a1[0]);
    System.out.println(a2[0]);
}

public static int[] duplicate(int[] theOriginal) {
   int[] theCopy = new int[theOriginal.length];
   for (int i = 0; i < theOriginal.length; i++)
      theCopy[i] = theOriginal[i];
  return theCopy;
}
```

```
with: a1 = duplicate(a2);
10
1
99
1
with: a1 = (int[]) a2.clone();
10
1
99
1
with:  a1 = a2;
 0
1
99
99
```

Output

Array Parameters

- You can also define a method that **takes in an entire array as input**
  - In such cases, the type name of the formal parameter is Base_Type[] (e.g., int[])
  - An array of any length can be passed
  - **Changing an array element within the method changes the original object**

For example

```
Public static void doubleArrayElements (double[] a)
{
…
     for (int i =0; i < a.length; i++)
          a[i] = a[i]*2;
}

…
double[] b = new double[10];
Someclass.doubleArrayElements(b);
```

Command-Line Arguments (p. 359)

The signature of **main** is:

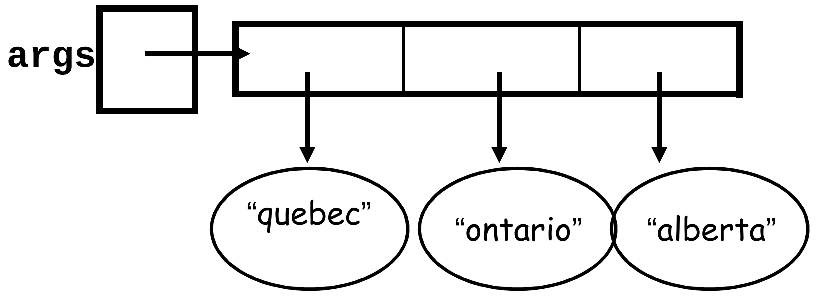
```
public static void main(String[] args)
```

The run time value of **arg** comes from the command-line arguments when the Java interpreter is invoked

Example:

```
> java MyProgram

> java MyProgram quebec ontario alberta
```



Example

```
public class NameTag
{
   public static void main (String[] args)
   {
      for (int i = 0; i < args.length; i++)
         System.out.println("argument[" + i + "] = " + args[i]);
   }
}
```

```
>
```

Command line

Methods That Return an Array

A method may also return an array

```
public static int[] incrementArray(int[] a, int increment)
{
  int[] temp = new int[a.length];
  int i;
  for (i = 0; i < a.length; i++)
    temp[i] = a[i] + increment;
  return temp;
}
```

Arrays are Reference Types

```
double[] a = new double[10];
double[] b = new double[10];

for(int i =0; i<a.length; i++)
   a[i] = i;
b=a
System.out.println(“a[2] = ” + a[2] + “b[2] = ” + b[2]);
a[2] = 42;
System.out.println(“a[2] = ” + a[2] + “b[2] = ” + b[2]);

Output:
a[2] = 2.0 b[2] = 2.0
a[2] = 42.0 b[2] = 42.0
```

Arrays are Reference Types cont'd…

- The assignment b = a **copies the memory address** from a to b so that the variable b contains the same memory address as the array variable a

- Unless you want two array variables to be two names for the same array, **you should not use the assignment '=' operator with arrays**

- Similarly, the '==' does not test if two arrays contain the same values. It **tests if two arrays are stored in the same location in memory**

Privacy Leaks with Array

If an accessor returns the contents of an array, special care must be taken

```
public double[] getArray()
{
   return anArray;//BAD!
}
```
privacy leak

Privacy Leaks with Array

The method should return a reference to a ***deep copy*** of the private array object

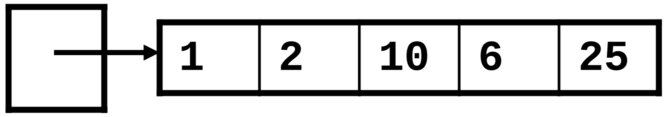
```
public double[] getArray()
{
  double[] temp = new double[count];
  for (int i = 0; i < count; i++)
    temp[i] = a[i];
  return temp;
}
```

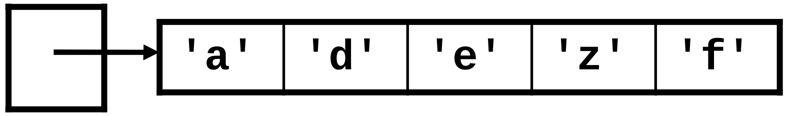
In this chapter, we will see…

1. Arrays **are** objects
2. **Arrays of objects**

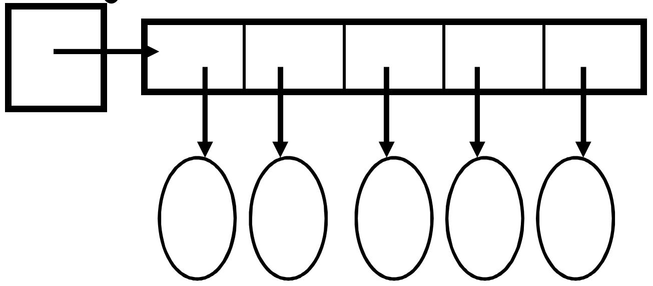
2- Arrays of Objects

- We can have arrays of primitive types





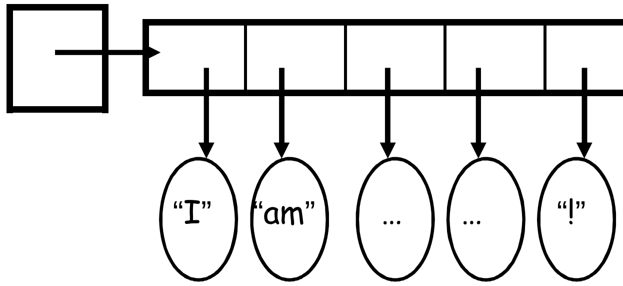
- We can have arrays of objects (more precisely, arrays of references to objects)



Example: Arrays of Strings

ex: 5 references to **`String`** objects

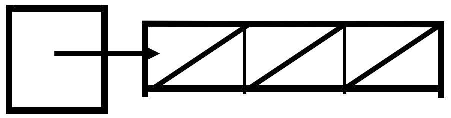
```
String[] words = {"I", "am", "very", "hungry", "!"};
```



Arrays of objects

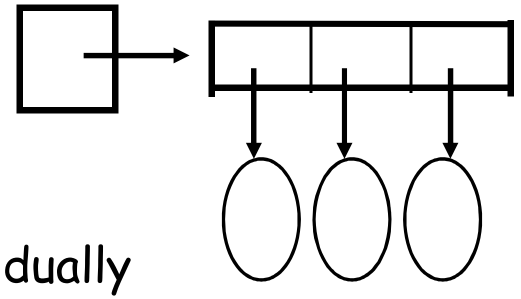
1. Create the array of references

```
Account[] bank = new Account[3];
```



2. Create each object

```
bank[0] = new Account("ted", 111, 100);
bank[1] = new Account("mary", 222, 500);
bank[2] = new Account("john", 999, 5);
```



Each object in the array must be created individually

3. Manipulate each object

```
for (int i = 0; i < bank.length; i++) {
    bank[i].deposit(100));
    System.out.print(bank[i].getBalance());
}
```

Example

Declare the class **Airplane** with **speed, nbPassengers, pilot**.

```
    private int speed;
    private int nbPassengers;
    private String pilot;
    public int getSpeed() {
        return speed;
    }
    public void setSpeed(int speed) {
        this.speed = speed;
    }
    public String toString()
    {
        return ("S:" + speed + " P:" + nbPassengers + " Pilot:" + pilot);
    }
```

Declare an array of 100 **Airplane** objects

```
        Airplane [] planes = new Airplane [100];
        for (int i = 0; i < planes.length; i++)
            planes[i] = new Airplane();

        for (int i = 0; i < planes.length; i++)
            System.out.println(planes[i].toString());
```

Example

write a method that takes an array of **`Airplane`** and returns the average speed of the **`Airplane`** . If the array is empty, return 0.

```
public static int avgSpeed(test[] p)
     {
          int sum = 0;
          for (int i = 0; i < p.length; i++)
               sum += p[i].speed;

          return((sum/p.length));
     }
```

THE END!