## Objectives

- In this session, you will learn to:
  - Declare and create arrays of primitive, class, or array types
  - Explain how to initialize the elements of an array
  - Determine the number of elements in an array
  - Create a multidimensional array
  - Write code to copy array values from one array to another
  - Define inheritance, polymorphism, overloading, overriding, and virtual method invocation
  - Define heterogeneous collections

## Arrays

- Declaring Arrays:
  - Group data objects of the same type.
  - An array is an object.
  - Declare arrays of primitive or class types:
    - `char` s[ ];
    - Point p[ ];
    - `char`[ ] s;
    - Point[ ] p;
  - The declaration of an array creates space for a reference.
  - Actual memory allocation is done dynamically either by a `new` statement or by an array initializer.

## Creating Arrays

- In order to create an array object:
  - Use the `new` keyword
  - An example to create and initialize a primitive (`char`) array:

```
public char[] createArray()
 {
        char[] s;
        s = new char[26];
        for ( int i=0; i<26; i++ )
        {
        s[i] = (char) ('A' + i);
        }
        return s;
 }
```

## Multidimensional Arrays

- A Multidimensional array is an array of arrays.

  For example:

```
int [] [] twoDim = new int [4] [];
twoDim[0] = new int [5];
twoDim[1] = new int[5];
```

  - The first call to `new` creates an object, an array that contains four elements. Each element is a null reference to an element of type array of `int`.
  - Each element must be initialized separately so that each element points to its array.

## Array Bounds

- All array subscripts begin at 0.
- The number of elements in an array is stored as part of the array object in the `length` attribute.
- The following code uses the `length` attribute to iterate on an array:

```
public void printElements(int[] list)
{
    for (int i = 0; i < list.length; i++)
    {
      System.out.println(list[i]);
    }
 }
```

## The Enhanced for Loop

- Java 2 Platform, Standard Edition (J2SE™) version 5.0 has introduced an enhanced `for` loop for iterating over arrays:

```
public void printElements(int[] list)
{
    for ( int element : list )
   {
       System.out.println(element);
   }
}
```

  - The `for` loop can be read as for each element in list do.

## Array Resizing

- You cannot resize an array.
- You can use the same reference variable to refer to an entirely new array, such as:

```
int[] myArray = new int[6];
myArray = new int[10];
```

  - In the preceding case, the first array is effectively lost unless another reference to it is retained elsewhere.

## Copying Arrays

- The Java programming language provides a special method in the `System` class, `arraycopy()`, to copy arrays.

  For example:

```
int myarray[] = {1,2,3,4,5,6}; // original array
int hold[] = {10,9,8,7,6,5,4,3,2,1}; // new larger array
System.arraycopy(myarray,0,hold,0,
myarray.length); // copy all of the myarray array to the hold array, starting with the 0th index
```

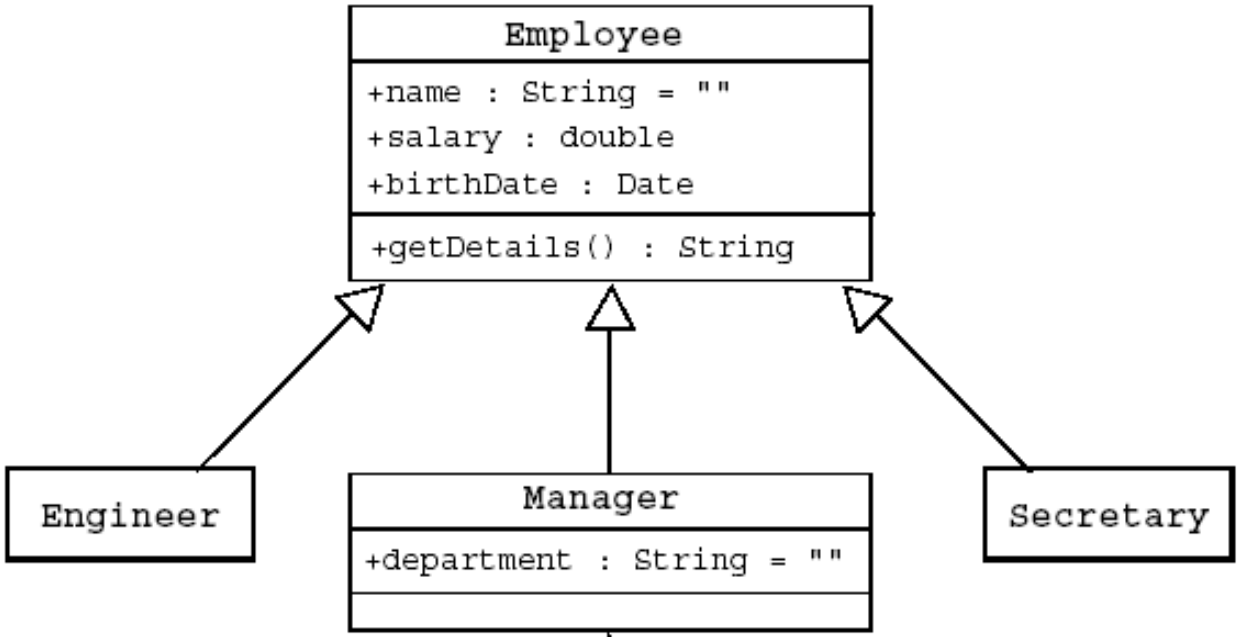
  - The contents of the array hold will be : 1,2,3,4,5,6,4,3,2,1.
  - The `System.arraycopy()` method copies references, not objects, when dealing with array of objects.

## Inheritance

- Inheritance means that a class derives a set of attributes and related behavior from a parent class.
- Benefits of Inheritance:
  - Reduces redundancy in code
  - Code can be easily maintained
  - Extends the functionality of an existing class

## Inheritance (Contd.)

- Single Inheritance
  - The subclasses are derived from one super class.
  - An example of single inheritance is as follows:



## Inheritance (Contd.)

- Java does not support multiple inheritance.
- Interfaces provide the benefits of multiple inheritance without drawbacks.
- Syntax of a Java class in order to implement inheritance is as follows:

```
<modifier> class <name> [extends
superclass>]
{   <declaration>*   }
```

## Access Control

- Variables and methods can be at one of the following four access levels:
  - `public`
  - `protected`
  - `default`
  - `private`
- Classes can be at the public or default levels.
- The default accessibility (if not specified explicitly), is package-friendly or package-private.

## Overriding Methods

- A subclass can modify behavior inherited from a parent class.
- Overridden methods cannot be less accessible.
- A subclass can create a method with different functionality than the parent's method but with the same:
  - Name
  - Return type
  - Argument list

## Overriding Methods (Contd.)

- A subclass method may invoke a superclass method using the `super` keyword:
  - The keyword `super` is used in a class to refer to its superclass.
  - The keyword `super` is used to refer to the members of superclass, both data attributes and methods.
  - Behavior invoked does not have to be in the superclass; it can be further up in the hierarchy.

## Overriding Methods (Contd.)

- Invoking Overridden Methods using `super` keyword:

```
public class Employee
{
   private String name;
   private double salary;
   private Date birthDate;
   public String getDetails()
   {
    return "Name: " + name +
    "\nSalary: " + salary;
   }
}
```

## Overriding Methods (Contd.)

```
public class Manager extends Employee
{
        private String department;
        public String getDetails() {
         // call parent method
        return super.getDetails()
        + "\nDepartment: " + department;
        }
 }
```

## Demonstration

Lets see how to create subclasses and call the constructor of the base class.

## Polymorphism

- Polymorphism is the ability to have many different forms; for example, the Manager class has access to methods from Employee class.
  - An object has only one form.
  - A reference variable can refer to objects of different forms.
  - Java programming language permits you to refer to an object with a variable of one of the parent class types.

    For example:

```
Employee e = new Manager(); // legal
```

## Virtual Method Invocation

- Virtual method invocation is performed as follows:

```
Employee e = new Manager();
e.getDetails();
```

  - Compile-time type and runtime type invocations have the following characteristics:
    - The method name must be a member of the declared variable type; in this case Employee has a method called getDetails.
    - The method implementation used is based on the runtime object's type; in this case the Manager class has an implementation of the getDetails method.

## Heterogeneous Collections

- Heterogeneous Collections:
  - Collections of objects with the same class type are called homogeneous collections. For example:

```
MyDate[] dates = new MyDate[2];
dates[0] = new MyDate(22, 12, 1964);
dates[1] = new MyDate(22, 7, 1964);
```

  - Collections of objects with different class types are called heterogeneous collections. For example:

```
Employee [] staff = new Employee[1024];
staff[0] = new Manager();
staff[1] = new Employee();
staff[2] = new Engineer();
```

## Summary

- In this session, you learned that:
  - Arrays are objects used to group data objects of the same type. Arrays can be of primitive or class type.
  - Arrays can be created by using the keyword `new`.
  - A multidimensional array is an array of arrays.
  - All array indices begin at 0. The number of elements in an array is stored as part of the array object in the `length` attribute.
  - An array once created can not be resized. However the same reference variable can be used to refer to an entirely new array.
  - The Java programming language permits a class to extend one other class i.e, single inheritance.

## Summary (Contd.)

- Variables and methods can be at one of the four access levels: `public`, `protected`, `default`, or `private`.
- Classes can be at the public or default level.
- The existing behavior of a base class can be modified by overriding the methods of the base class.
- A subclass method may invoke a superclass method using the `super` keyword.
- Polymorphism is the ability to have many different forms; for example, the Manager class (derived) has the access to methods from Employee class (base).
- Collections of objects with different class types are called heterogeneous collections.