

Software Engineering Laboratory

Course Organization

People

- Teachers:
 - Prof. Abir Das
 - Prof. Sourangshu Bhattacharya
- Teaching Assistants:
 - Owais Iqbal
 - V Nikhil Reddy
 - Uppada Vishnu
 - Nilesh Laad
 - Aakash Naik
 - Omprakash Chakraborty

Course Conduction

- 3 class tests
 - Time bound.
 - No discussion.
 - Zero in the test if found communicating.
- 6 – 7 assignments
 - Discussion allowed
 - Submission will be checked for plagiarism
 - Zero in assignment if found to be source or benefactor of plagiarism.
- Grading: 60 assignment + 40 Tests

Course Conduction

- Class – all discussions and announcement will be done on Teams.
 - Must join the Teams meeting – attendance will be taken.
- Assignments
 - Assignments will be floated on moodle.
 - Submission to be made on moodle before deadline.
 - Individually done if not explicitly mentioned a group assignment.
 - TAs will be assigned to students for helping and grading with assignments.
- Class Tests:
 - Conducted on moodle online.

Schedule

Dates	Activity	Topic
06 January 2021	Introduction + Assignment 0	Java Programming
13 January 2021	Assignment 1	C++ Programming
20 January 2021	Assignment 2	Python Programming (GUI)
27 January 2021	Class test 1	C++ and Java Programming
03 February 2021	Assignment 3	Python Programming (Data Science)
10 February 2021	Discussion + floating of projects	
17 February 2021	Assignment 4	SRS, SA&D, UML

Schedule

Dates	Activity	Topic
24 February 2021	Class test 2	Python Programming
03 March 2021	Spring Break	
10 March 2021	Assignment 5	Implementation
17 March 2021	Assignment 6	Testing
24 March 2021	Discussion	
31 March 2021	Class Test 3	
07 April 2021	Doubt clearing	
14 April 2021	Marks finalization	

Crash course on Java

Content

- “Hello World” program example and running
- Java language Syntax
- Inheritance

Java programming Language

- Some buzzwords for Java
 - “Write Once, Run Anywhere”
 - Simple
 - **Object oriented**
 - Distributed
 - Multithreaded
 - Dynamic
 - **Architecture neutral**
 - Portable
 - High performance
 - Robust
 - Secure

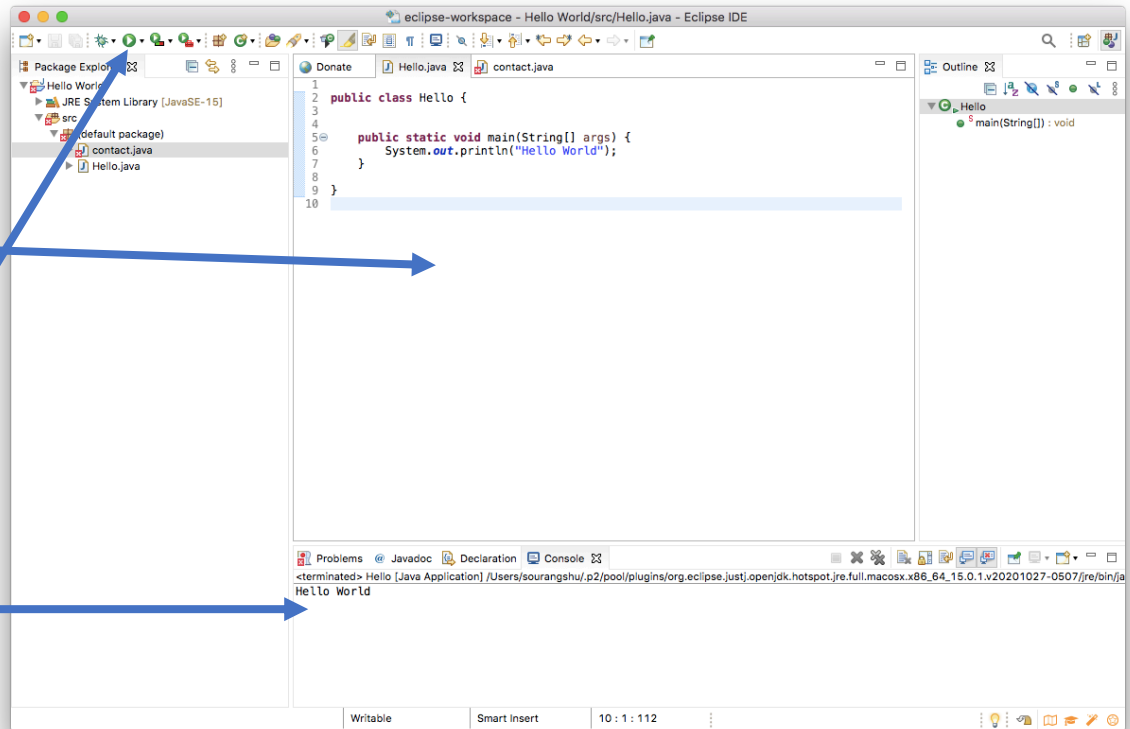
Example: Hello World Program

```
public class HelloWorld {  
  
    public static void main(String[] args) {  
        System.out.println("Hello World!");  
    }  
}
```

- Everything is in a class
- One file, one public class
- In the runnable public class:
 - `public static void main(String [] args)`

Running Java Progs.

- Download eclipse: <https://www.eclipse.org/downloads/>
- Create new project
- Create new java file
- Write your program here
- Hit the run button
- Output is here



Primitive Data Types

- **Primitive Data Types:** byte, short, int, long, float, double, boolean, char

- **Arrays** are also a class

```
long a [] = new long[5];
```

- You can get the length by visiting the length field of array object a, like this: **a.length**

- **String** class is very commonly used to represents character strings, for example

```
String s1 = "Hello ", s2 = "World!";  
String s3 = s1 + s2;
```

Operators (same as C/C++)

- ++,-- Auto increment/decrement
- +,- Unary plus/minus
- *,/ Multiplication/division
- % Modulus
- +,- Addition/subtraction

Declaring Variables

```
int n = 1;
```

```
char ch = 'A' ;
```

```
String s = "Hello";
```

```
Long L = new Long(100000) ;
```

```
boolean done = false;
```

```
final double pi = 3.14159265358979323846;
```

```
Employee joe = new Employee() ;
```

```
char [] a = new char[3] ;
```

```
Vector v = new Vector() ;
```

Compared with C/C++

- Java has no:
 - pointers
 - typedef
 - preprocessor
 - struct
 - unions
 - multiple inheritance
 - goto
 - operator overloading
 - malloc
 - ...

Declaring a class

- Class name
- Constructor
- Fields
- methods

```
public class Person {  
    //fields (or 'data members' in C++)  
    private String name;  
    private int age;  
    //constructor method  
    public Person(){  
        this.name="Unknown person";  
        this.age = 0;  
    }  
    //methods (or 'functions' in C++)  
    public String getName(){  
        return this.name;  
    }  
    public int getAge(){  
        return this.age;  
    }  
    //Optional main method, which is a main execution entry point  
    public static void main(String args[]){  
        //creating a new object that is an instance of the class Person  
        Person p = new Person();  
        //calling the method of p instance  
        //in this case, name will be "Unknown person"  
        String name = p.getName();  
        //print name  
        System.out.println(name);  
    }  
}
```

Inheritance in Java

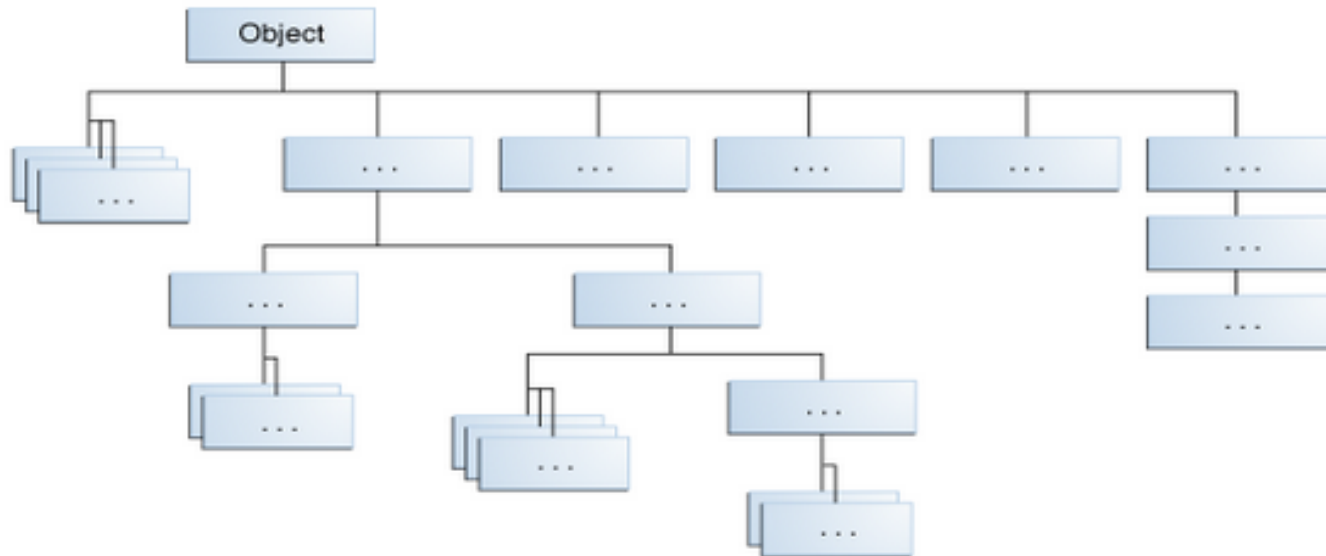
- Java classes can be *derived* from other classes, thereby *inheriting* fields and methods from those classes.

```
public class Animal{
    private int age;
    public void move(){
        System.out.print("The Animal is moving");
    };
}

class Cat extends Animal{
    //a method in the sub-class
    public void meow(){
        System.out.print("The Cat is meowing");
    };
}

class Dog extends Animal{
    //a method in the sub-class
    public void bark(){
        System.out.print("The Dog is barking.");
    };
}
```

Common Root: Object



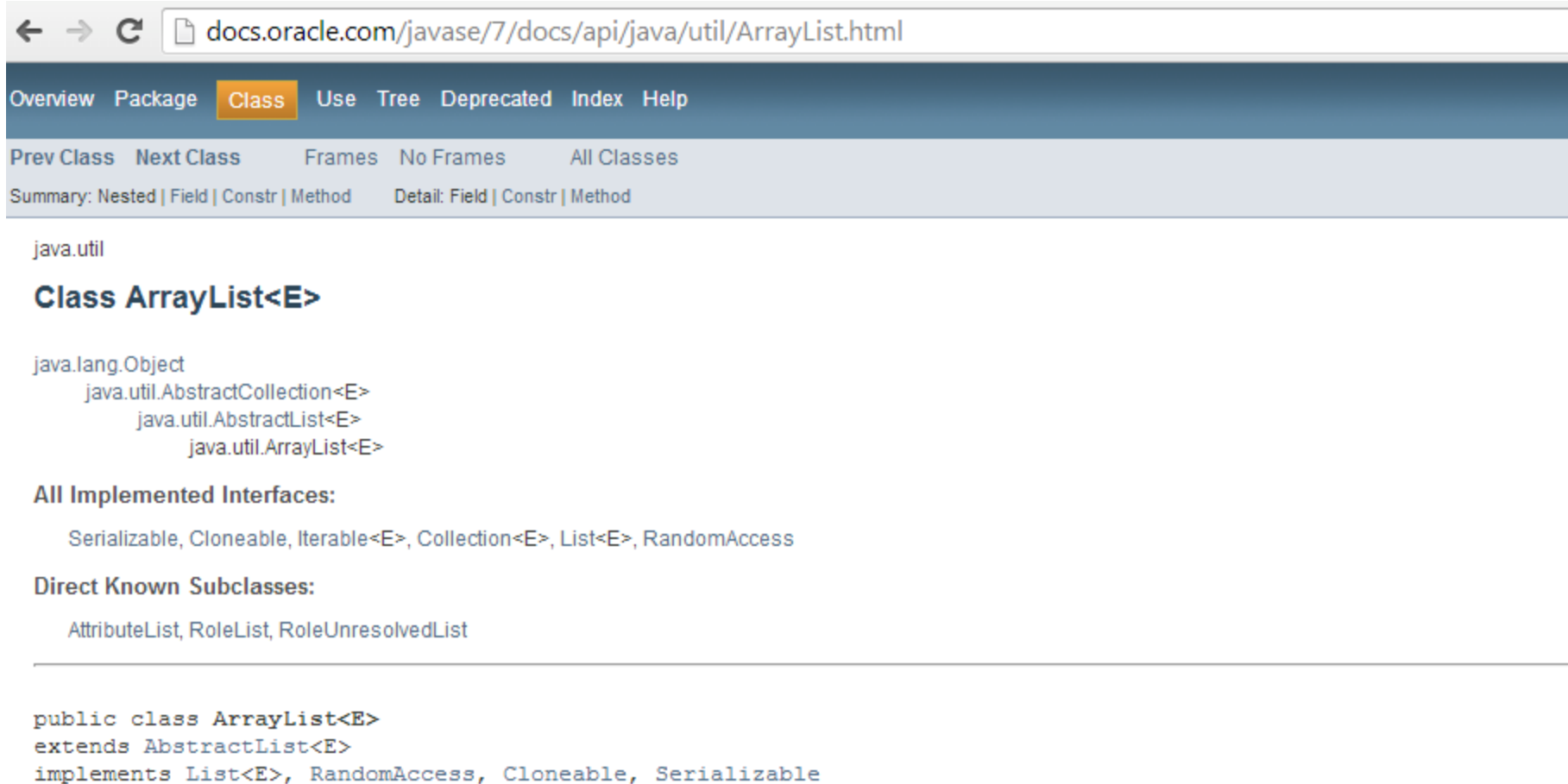
Interface

```
public interface Animal {  
    public void move();  
    public void eat();  
}  
  
class Dog implements Animal  
{  
    public void move() {  
        System.out.println("The Dog is moving.");  
    }  
    public void eat() {  
        System.out.println("The Dog is eating.");  
    }  
}  
  
class Cat implements Animal  
{  
    public void move() {  
        System.out.println("The Dog is moving.");  
    }  
    public void eat() {  
        System.out.println("The Dog is eating.");  
    }  
}
```

“Multiple Inheritance”

```
public interface Bird {  
    public void fly();  
}  
  
interface MythologicalCreature{  
    //Mythological Creatures can speak human languages  
    public void speak();  
}  
  
class Horse {  
    public void run(){  
        System.out.println("The Horse is running");  
    }  
}  
  
class Pegasus extends Horse implements Bird, MythologicalCreature{  
    public void fly(){  
        System.out.println("The Pegasus is running");  
    }  
    public void speak(){  
        System.out.println("The Pegasus is speaking human languages");  
    }  
}
```

A Real World Example: ArrayList



The screenshot shows the Oracle Java API documentation for the `ArrayList` class. The browser address bar displays `docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html`. The navigation bar includes links for Overview, Package, Class (highlighted), Use, Tree, Deprecated, Index, and Help. Below this, there are links for Prev Class, Next Class, Frames, No Frames, and All Classes. The main content area shows the package `java.util` and the class `Class ArrayList<E>`. It lists the inheritance hierarchy: `java.lang.Object`, `java.util.AbstractCollection<E>`, `java.util.AbstractList<E>`, and `java.util.ArrayList<E>`. It also lists all implemented interfaces: `Serializable`, `Cloneable`, `Iterable<E>`, `Collection<E>`, `List<E>`, and `RandomAccess`. Direct known subclasses are listed as `AttributeList`, `RoleList`, and `RoleUnresolvedList`. At the bottom, the class declaration is shown: `public class ArrayList<E> extends AbstractList<E> implements List<E>, RandomAccess, Cloneable, Serializable`.

← → ↻ docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html

Overview Package **Class** Use Tree Deprecated Index Help

Prev Class Next Class Frames No Frames All Classes

Summary: Nested | Field | Constr | Method Detail: Field | Constr | Method

java.util

Class ArrayList<E>

java.lang.Object
 java.util.AbstractCollection<E>
 java.util.AbstractList<E>
 java.util.ArrayList<E>

All Implemented Interfaces:

Serializable, Cloneable, Iterable<E>, Collection<E>, List<E>, RandomAccess

Direct Known Subclasses:

AttributeList, RoleList, RoleUnresolvedList

```
public class ArrayList<E>
    extends AbstractList<E>
    implements List<E>, RandomAccess, Cloneable, Serializable
```

Example problem

- You have a set of acquaintances.
- The acquaintances are of types: personal friends and professional friends.
- For all acquaintances, you need to remember the name, mobile number, and e-mail.
- For personal friends, you would be interested to remember the birth day.
- For professional friends, you would like to remember the specific common professional interests (100 chars max).
- You should be able to perform the following:
 - Create and delete various types of acquaintances.
 - Display the entire list of acquaintances.

Program

```
public class ContactList {  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        Friend fl[] = new Friend[5];  
        Scanner myObj = new Scanner(System.in); // Create a Scanner object  
  
        String myinput=myObj.nextLine();  
        int i=0;  
        //if(myinput.equals("sdf")) System.out.println(myinput);  
        //ContactList cl = new ContactList();  
        while(!myinput.equals("quit")) {  
            fl[i] = CreateFriend(myObj);  
            //fl[i].printobj();  
            myinput=myObj.nextLine();  
            i++;  
        }  
        int n=i;  
        for(i=0;i<n;i++) {  
            fl[i].printobj();  
        }  
    }  
}
```


Program

```
static Friend CreateFriend(Scanner myObj) {  
    //Scanner myObj = new Scanner(System.in); // Create a Scanner object  
    System.out.print("Enter the type: ");  
    String mytype=myObj.nextLine();  
    if(mytype.equals("personal")) return new PersonalFriend(myObj);  
    if(mytype.equals("professional")) return new ProfessionalFriend(myObj);  
    return null;  
}  
}
```

Program

```
class Friend {
    String name;
    String mobileno;
    String emailid;

    public Friend(Scanner myObj) {
        // TODO Auto-generated constructor stub
        //name=null;
        //mobileno=null;
        //emailid=null;
        //Scanner myObj = new Scanner(System.in); // Create a Scanner object
        System.out.print("Enter the name: ");
        this.name=myObj.nextLine();
        System.out.print("Enter the mobileno: ");
        this.mobileno=myObj.nextLine();
        System.out.print("Enter the eamilid: ");
        this.emailid=myObj.nextLine();
    }

    void printobj() {
        System.out.println("Name: "+name);
        System.out.println("Mobile no: "+mobileno);
        System.out.println("Email: "+emailid);
    }
}
```

Program

```
class PersonalFriend extends Friend {  
    String birthday;  
  
    public PersonalFriend(Scanner myObj) {  
        // TODO Auto-generated constructor stub  
        super(myObj);  
        //birthday = new Date();  
        System.out.print("Enter the Birthday: ");  
        this.birthday=myObj.nextLine();  
    }  
    void printobj() {  
        super.printobj();  
        System.out.println("Birthday: "+birthday);  
    }  
}
```

Program

```
class ProfessionalFriend extends Friend {
    String interests;

    public ProfessionalFriend(Scanner myObj) {
        // TODO Auto-generated constructor stub
        super(myObj);
        //birthday = new Date();
        System.out.print("Enter the Common Interests: ");
        this.interests=myObj.nextLine();
    }

    void printobj() {
        super.printobj();
        System.out.println("Common Interests: "+interests);
    }
}
```