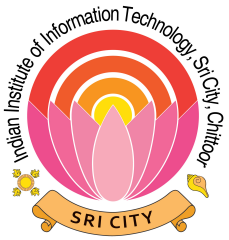


Monsoon 2019

Inheritance (2) and Collections

Object Oriented Programming

by

Dr. Rajendra Prasath

Indian Institute of Information Technology

Sri City – 517 646, Andhra Pradesh, India

Recap: Objects in JAVA ?

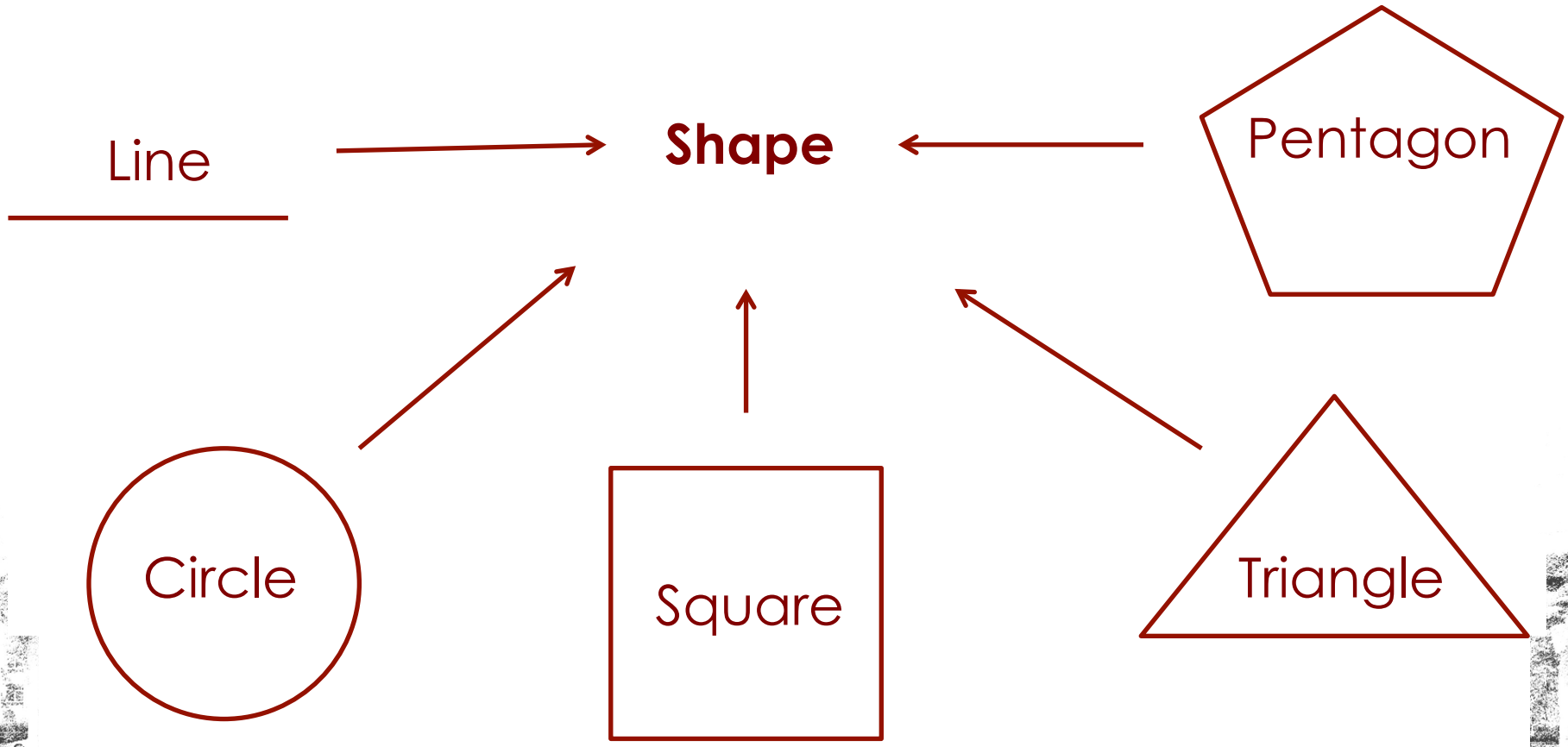
- An entity that has **state** and **behaviour** is known as an object
  - **Examples:** Chair, bike, marker, pen, table, car etc
  - It can be physical or logical
- An object has three characteristics:
  - **State:** represents data (value) of an object
  - **Behaviour:** represents the behaviour (functionality) of an object such as deposit, withdraw and so on
  - **Identity (Internally used):**
    - Signature (unique) of the object
    - Object identity is typically implemented via a unique ID
    - The value of the ID is not visible to the external user
    - But, Internally by JVM to identify each object uniquely

Recap: Anonymous Inner Class

```
Interface Message{
        String welcome();
}
public class CodeTester {
        public void showMessage(Message m) {
                System.out.println(m.welcome());
        }
        Public static void main(String args[]) {
                CodeTester ct = new CodeTester();
                ct.showMessage(new Message() {
                        public String welcome() {
                                return "Welcome Folks!!";
                        }
                });
        }
}
```

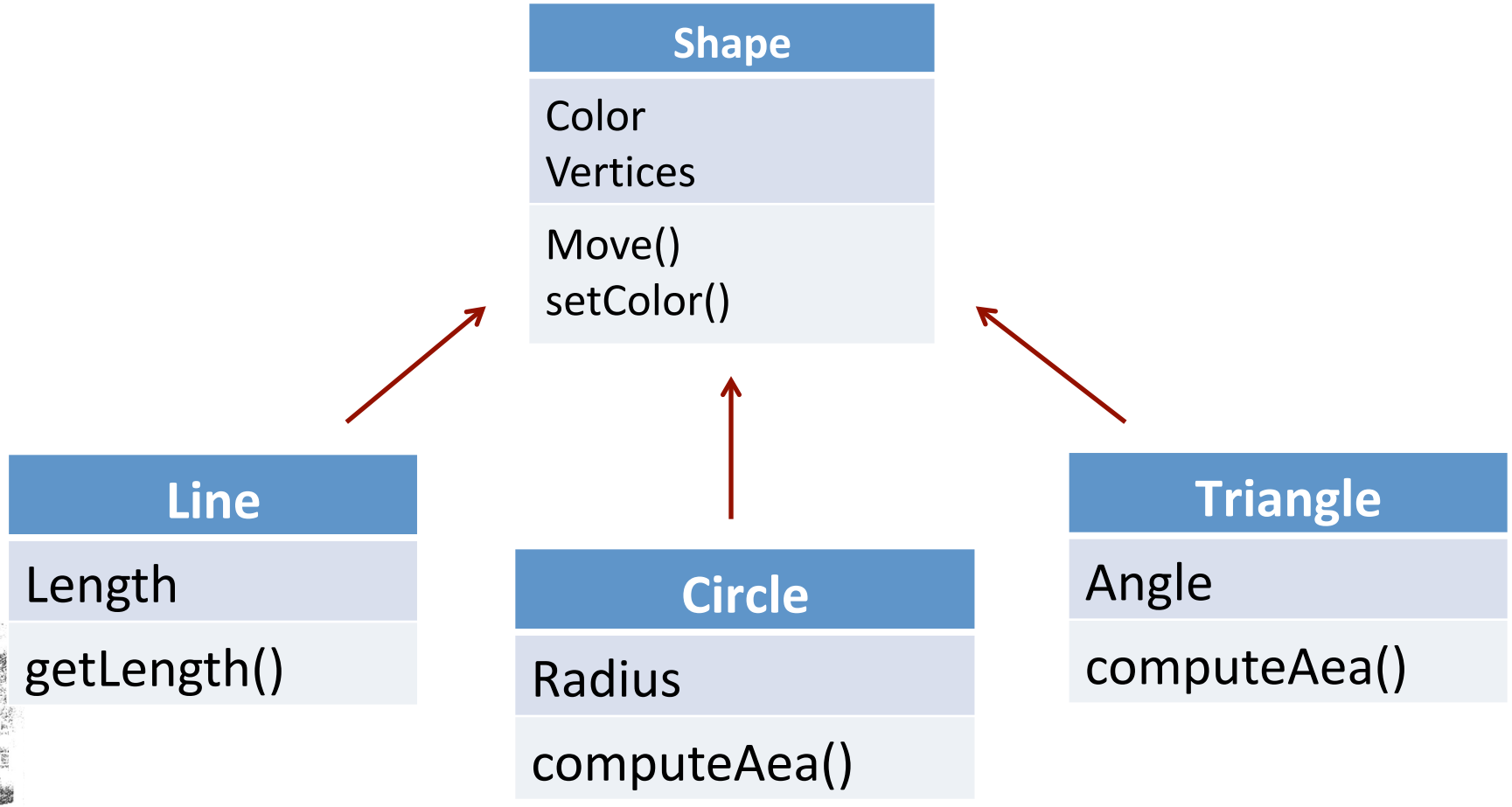
Recap: Inheritance

- Let us look at another example:



Recap: Generalization

- Create a Class and inherit its properties



Recap: IS-A Relationship

- IS-A is a way of saying: **This object is a type of that object**
- Let us look at the following example:

```
public class Vehicle {
}
public class TwoWheeler extends Vehicle {
}

public class FourWheeler extends Vehicle {
}

public class Car extends FourWheeler {
}
```

Superclass

Subclass

Subclass

Subclass of both FourWheeler and Vehicle classes

IS-A relationship

TwoWheeler IS-A Vehicle
FourWheeler IS-A **Vehicle**

Car IS-A **FourWheeler**

So
Car is a **Vehicle**

Inheritance – An Example

- Let us look at the following example:

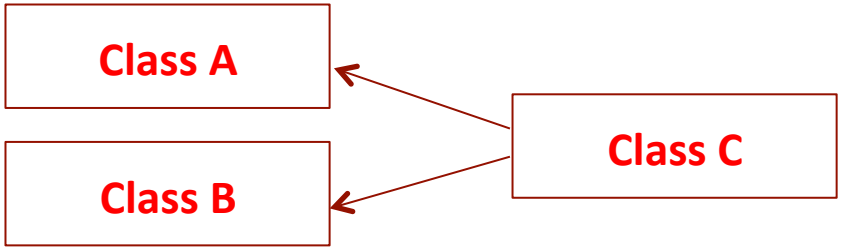
```
Class Parent {
        public void p1() {
                System.out.println("Parent Method!!");
        }
}

Class Child extends Parent{
        public void c1() {
                System.out.println("Child Method!");
        }

        public static void main(String[] args) {
                Child ch = new Child();
                ch.c1();
                ch.p1();
        }
}
```

Types of Inheritance – 5 types

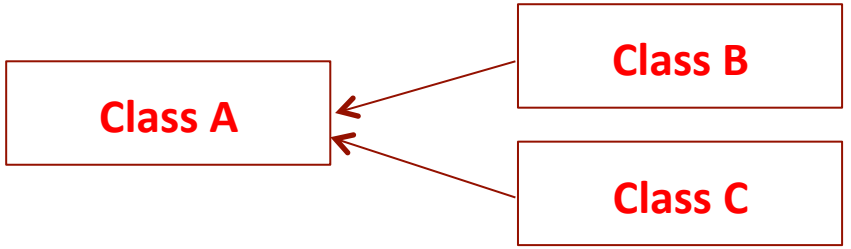
- Single Level inheritance

- Multi Level inheritance

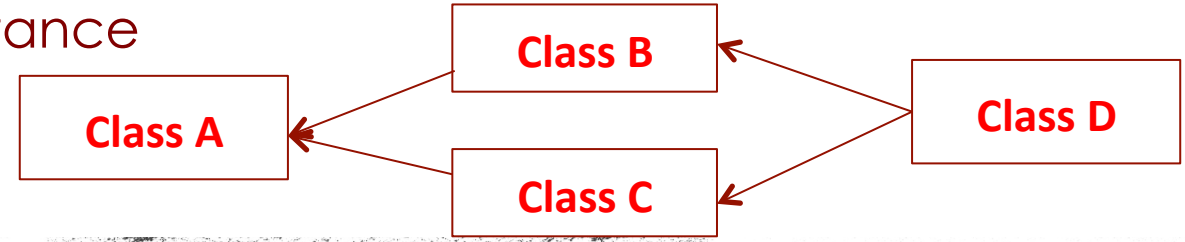
- Multiple inheritance



- Hierarchical inheritance



- Hybrid inheritance



Single Inheritance - example

```
Class A {
        int data = 10;
}

Class B extends A {
        public void display() {
                System.out.println("Data is:" + data);
        }
        public static void main(String[] args) {
                B b = new B();
                b.display();
        }
}
```

Multilevel Inheritance - example

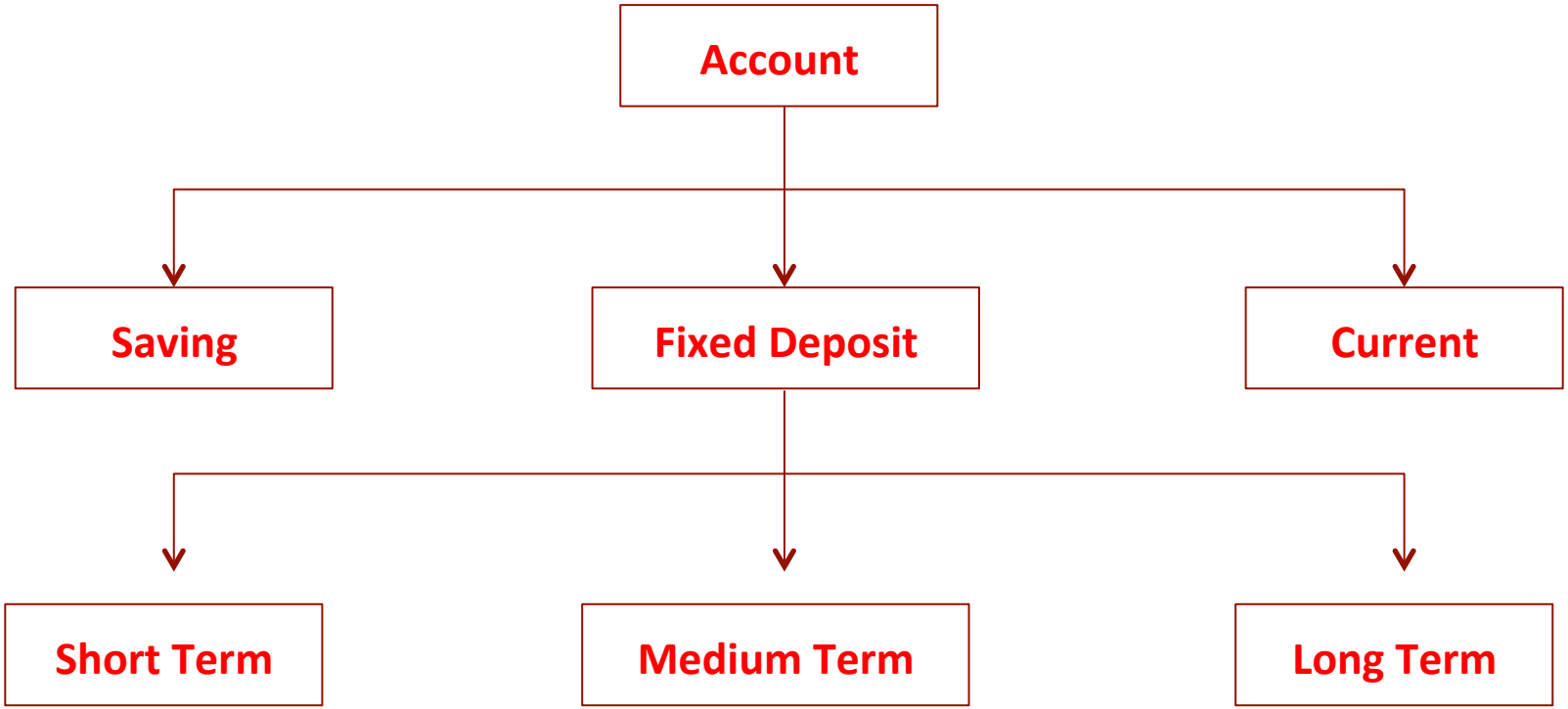
```
Class A {
        int data = 10;
}

Class B extends A {
        float salary = 30000.;
}

Class C extends B {
        public void display() {
                System.out.println("Data is = " + data);
                System.out.println("Salary is = " + salary);
        }
        public static void main(String[] args) {
                C c = new C();
                c.display();
        }
}
```

Multilevel Inheritance-Employee

```
Class Employee {
        float total = 0.0, salary = 10000.;
}

Class HRA extends Employee {
        float hra = 3000.;
}

Class DA extends HRA {
        float da = 5000.;
}
Class Income extends DA {
        float bonus = 3000.;
        public static void main(String[] args) {
                Income net = new Income();
                net.total = net.salary + net.hra + net.da + net.bonus;
                System.out.println("Salary is = " + net.total);
        }
}
```

Hierarchical Inheritance

✧ Real time example



Hierarchical - Example

```
Class A {
        int data = 10;
}

Class B extends A {
}

Class C extends A {
}

Class D extends A {
        public static void main(String[] args) {
                B b = new B();
                C c = new C();
                D d = new D();
                System.out.println("Salary is = " + b.data);
                System.out.println("Salary is = " + c.data);
                System.out.println("Salary is = " + d.data);
        }
}
```

Multiple Inheritance – Recap

- Let us look at the following Example:

```
class A {
        void display() {
                System.out.println("Hello (A)");
        }
}
class B {
        void display() {
                System.out.println("Welcome (B)");
        }
}
class C extends A, B {
        public static void main(String[] args) {
                C c = new C();
                c.display();
        }
}
```

Hybrid Inheritance

- Any combination of previous three inheritance **(single, hierarchical and multi level)** is called as hybrid inheritance

- In simple terms, one can say that Hybrid inheritance is a combination of Single and Multiple inheritance

- A hybrid inheritance can be achieved in the java using interfaces

Important Points for Inheritance

- **Inheritance in JAVA:**
  - One derived class can extends only one base class because java programming does not support multiple inheritance through **the concept of classes**, but it can be supported through **the concept of Interface**.
  - To develop any inheritance application, first create an object of bottom most derived class, but not for top most base class.
  - When we create an object of bottom most derived class, first we get the memory space for the data members of top most base class, and then we get the memory space for data member of other bottom most derived class.
  - Bottom most derived class contains logical appearance for the data members of all top most base classes.

Some More Salient Points

- **Inheritance in JAVA:**
  - If you do not want to give the features of base class to the derived class then the definition of the base class must be **preceded by final**
    - **Final base classes are not reusable or not inheritable**.
  - If you do not want to give some of the features of the base class to its derived class then such features of base class must be declared as private
    - Private features of base class are not inheritable or accessible in the derived class.
  - **Scope of base class object:** An object of the base class can contain details about features of same class but an object of base class never contains the details about special features of its derived class
  - For each and every class in java there exists an implicit predefined super class called java.lang.Object. Because it provides garbage collection facilities to its sub classes for
    - collecting un-used memory space
    - improved the performance of java application

Having Same Data Member(s)

- Let us look at the following example:

```
class Parent {
        int value = 1000;
        Parent() {
                System.out.println("Parent Constructor");
        }
}
class Child extends Parent {
        int value = 10;
        Child() {
                System.out.println("Child Constructor");
        }
}
```

Same Data Member – main()

- **main() method**

```
class Test {
    public static void main(String[] args) {
        Child obj = new Child();
        System.out.println("Reference of Child Type :" + obj.value);

        // Similar to "Parent par = new Child()"
        Parent par = obj;

        // obj will access the value of the variable of parent class
        System.out.println("Reference of Parent Type : "+ par.value);
    }
}
```

The Reference Variable of the Parent class is capable to hold its Object Reference as well as its child Object Reference.

Analysis

- **Run-time Polymorphism:**
  - If a parent reference variable is holding the reference of the child class and we have the "value" variable in both the parent and child class, it will refer to the parent class "value" variable.
  - The reference holding the child class object reference will not be able to access the members (functions or variables) of the child class.
    - It is because compiler uses special **run-time polymorphism mechanism only for methods.**
  - It is possible to access child data members using parent pointer with typecasting.

Learning

- **Data Hiding:**
  - Whenever a parent class and a child class are having same data members, then this concept is known as "Data Hiding"
- **Method Overriding:**
  - Whenever a parent class and a child class are having same methods, then this concept is known as Method Overriding

- Always data members of parent class is inherited by the keyword **super**
- If any non-static method of a class is made as **final** then it can not be overridden by the child class

Inheritance and Constructors

- Constructor of the base with no arguments gets automatically called in the derived class constructor

```
Class Parent {
        Parent() {
                System.out.println("Parent Class Constructor");
        }
}
Class Child extends Parent {
        Child() {
                System.out.println("Child Class Constructor");
        }
}
public class CodeTester {
        public static void main(String[] args) {
                Child child = new Child();
        }
}
```

Java Collections

- A collection is a group of individual objects represented as a single unit.
- Java provides Collection Framework which defines several classes and interfaces to represent a group of objects as a single unit.
- Two main "root" interfaces
  - **Collection Interface** (java.util.Collection)
    - Set
    - List
    - Queue
    - Deque
  - **Map Interface** (java.util.Map)
    - Map
    - SortedMap

Word Count - A Practical Appln

- Given n documents
- How to find the total count of words present in these n documents?
- Can we use the util collection?

Look at 3 documents

d_1- **Darjeeling** is a city and a municipality in the Indian state of West Bengal. It is located in the Lesser Himalayas at an elevation of 6,700 feet

d_2- **Darjeeling** is noted for its tea industry, its views of Kangchenjunga, the world's third-highest mountain, and the **Darjeeling** Himalayan Railway, a UNESCO World Heritage Site

d_3- **Darjeeling** is the headquarters of the **Darjeeling** District which has a partially autonomous status within the state of West Bengal. It is also a tourist destination in India

Unique words and Counts?

d_1

2 the
2 of
2 is
2 in
2 a
1 state
1 municipality
1 located
1 feet
1 elevation
1 city
1 at
1 and
1 an
1 West
1 Lesser
1 It
1 Indian
1 Himalayas
1 Darjeeling
1 Bengal
1 6,700

d_2

2 the
2 its
2 Darjeeling
1 world's
1 views
1 third-highest
1 tea
1 of
1 noted
1 mountain
1 is
1 industry,
1 for
1 and
1 a
1 World
1 UNESCO
1 Site
1 Railway
1 Kangchenjunga
1 Himalayan
1 Heritag

d_3

3 the
2 of
2 is
2 a
2 Darjeeling
1 within
1 which
1 tourist
1 status
1 state
1 partially
1 in
1 headquarters
1 has
1 destination
1 autonomous
1 also
1 West
1 It
1 India
1 District
1 Bengal

Documents – Words / Terms*

✧ How to construct Terms - documents

Doc ID	Terms	# Words
d_1	6,700 (1), Bengal. (1), Darjeeling (1), Himalayas (1), Indian (1), It (1), Lesser (1), West (1), a (2), an (1), and (1), at (1), city (1), elevation (1), feet (1), in (2), is (2), located (1), municipality (1), of (2), state (1), the (2),	22
d_2	Darjeeling (2), Heritage (1), Himalayan (1), Kangchenjunga, (1), Railway, (1), Site (1), UNESCO (1), World (1), a (1), and (1), for (1), industry, (1), is (1), its (2), mountain, (1), noted (1), of (1), tea (1), the (2), third-highest (1), views (1), world's (1),	22
d_3	Bengal. (1), Darjeeling (2), District (1), India (1), It (1), West (1), a (2), also (1), autonomous (1), destination (1), has (1), headquarters (1), in (1), is (2), of (2), partially (1), state (1), status (1), the (3), tourist (1), which (1), within (1),	22

NOTE: "**Words**" and "**Terms**" are interchangeably used throughout the course

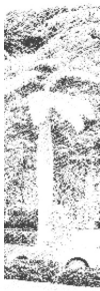
Terms - Documents

Terms	d_1	d_2	d_3	. . .	d_n
the	2	2	3	. . .	0
a	2	1	2	. . .	1
Darjeeling	1	2	2	. . .	0
is	2	1	2	. . .	0
of	2	1	2	. . .	0
in	2	0	0	. . .	1
and	1	1	0	. . .	0
Bengal	1	0	1	. . .	0
It	1	0	1	. . .	0
Its	0	2	0	. . .	2
state	1	0	1	. . .	0
West	1	0	1	. . .	1

NOTE: "**Words**" and "**Terms**" are interchangeably used throughout the course

Exercise - 7

- **Apply the concepts of Inheritance for**
  - Developing a Banking System
  - Design class diagrams and relationship among the objects
  - Simplify the tasks in mobile payments
  - Define access control mechanism that provides security features for such an application



Assignments / Penalties

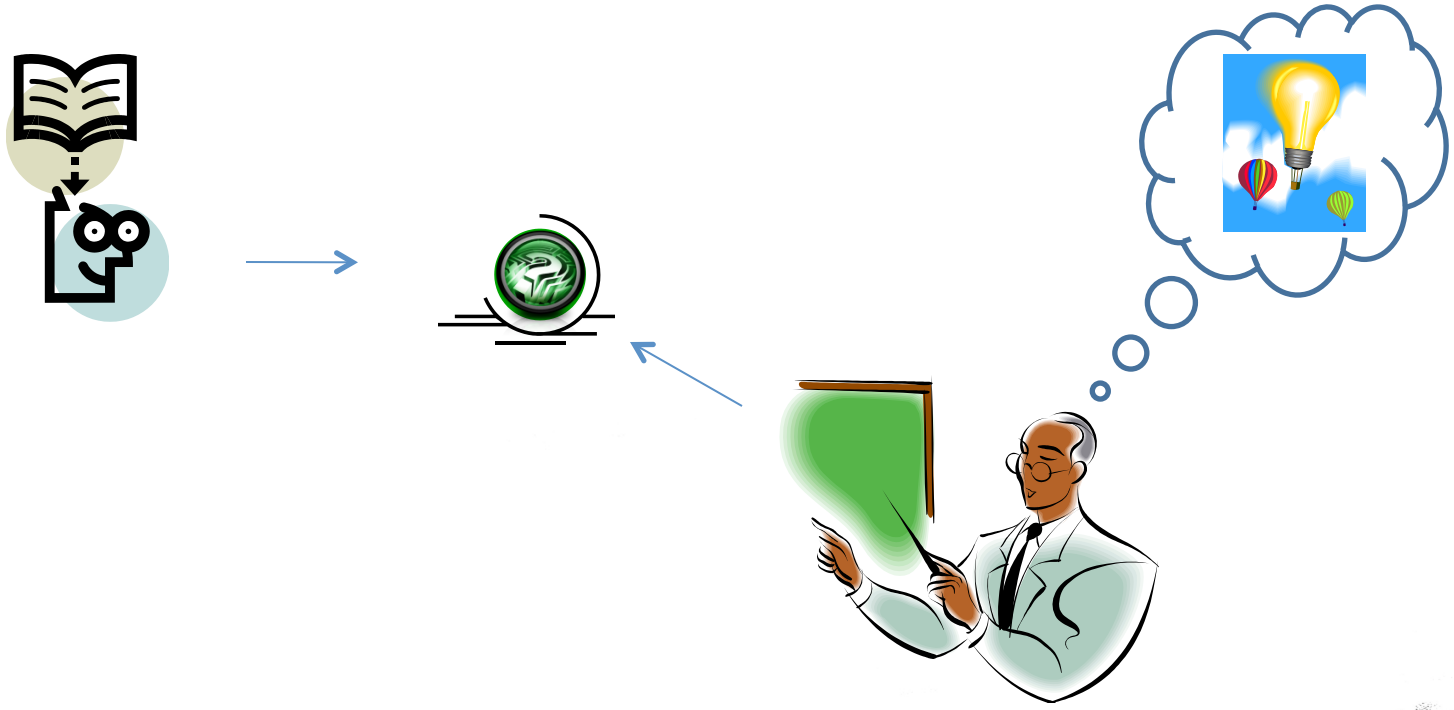
- Every Student is expected to complete the assignments and strictly follow a fair Academic Code of Conduct to avoid severe penalties

- Penalties would be heavy for those who involve in:
  - **Copy and Pasting** the code
  - **Plagiarism** (copied from your neighbor or friend – in this case, both will get "0" marks for that specific take home assignments)
  - If the candidate is **unable to explain his own solution**, it would be considered as a "copied case" !!
  - **Any other unfair means** of completing the assignments

Assistance

- You may post your questions to me at any time
- You may meet me in person on available time or with an appointment
- You may leave me an email any time
  (email is the best way to reach me faster)

Thanks …



… Questions ???