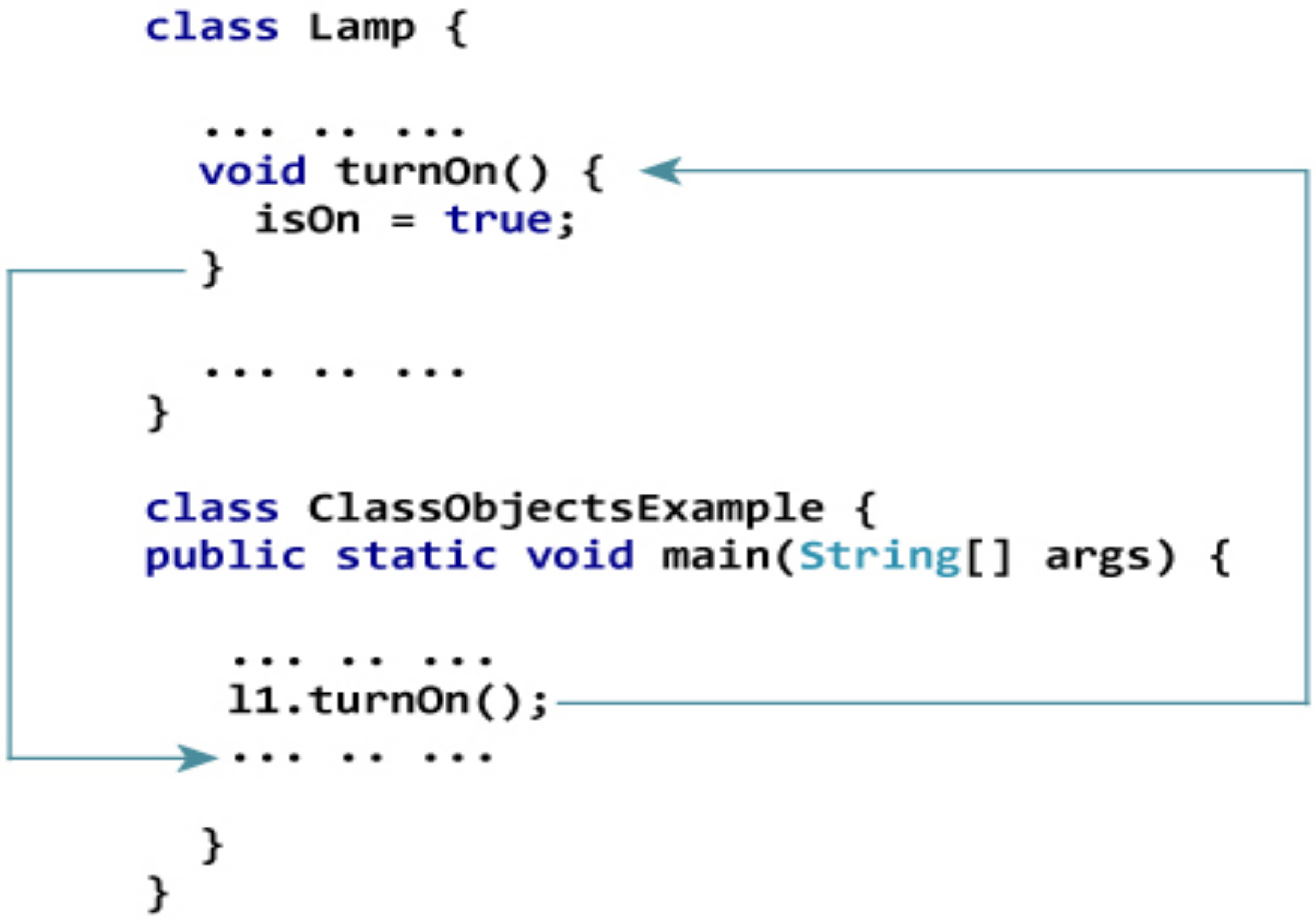
- These objects share two characteristics:
  - state (fields, data, value)
  - behavior (methods)
- For example,
- Lamp is an object
  State: on or off
  Behavior: turn on or turn off
- Bicycle is an object
  States: current gear, two wheels, number of gear, etc
  Behavior: braking, accelerating, changing gears, etc

```
class Lamp
{ // instance variable
    private boolean isOn;
// method
   public void turnOn()
    {
       isOn = true;
     }
// method
   public void turnOff()
     {
        isOn = false;
      }
 } //end of class
```



```
class Lamp {

  ... .. ...
  void turnOn() {
    isOn = true;
  }

  ... .. ...
}

class ClassObjectsExample {
public static void main(String[] args) {

   ... .. ...
   l1.turnOn();
   ... .. ...

  }
}
```

```
public class MyClass
{
  int x = 5;
Int m,n;
Public void display()
{
}
 public static void main(String[] args) //main method
    {
      int y = 0;
      MyClass myObj = new MyClass();
      y = myObj.x;// direct object reference
      muObj.display();
      System.out.println(y);
    }

}
//output
5
```

```
Public Class Student
{
  String Name, class, section;
Int RegNo, roll_no;

Public register_sem()
 {
}
AttendClass()
{
}
Access specifier return value GiveExam(parameters)
{
}
Public static void main()
{
}
}// end of class student
Class course
{
}
Class faculty
{
}
```

```
public class MyClass
{
 int x = 5;
  public static void main(String[] args)
    {
       MyClass myObj1 = new MyClass(); // Object 1
       MyClass myObj2 = new MyClass(); // Object 2
       System.out.println(myObj1.x);
       System.out.println(myObj2.x);
    }
}
```

MyClass.java

Myclass.java

myclass.java

- Create a class for your vehicle
- Create a class for your favourite gadget

- --- Class ----
- {
- //variables
- //methods()
- }

```
Public class Car
 {
    //attributes
    brand =“Maruti Suzuki”
    Model= “Celerio”
    Color = “White”
// methods
  public void Start()
  {
   }
Public GreaChange()
  {
  }
}
```

Initialization through method

```
class Student
{
  private int id;
   String name;
  void insertRecord(int r, String n)
   {
     id=r;
     name=n;
   }
 int getId()
   {
      return id;
    }
Void displayIfo()
{
 System.out.println(id+" "+name)
}
}
```

Initialization through method

```
class TestStudent2
 {
 public static void main(String args[])
 {
    Student s1=new Student();
    s1.id=101;
   s1.name="Priya";
              S1. insertRecord(101,"Priya");
              S1.displayInfo();
S1.id // not allowed

   int id1 = s1.getId(); //allowed
    System.out.println(id1);
 // System.out.println(s1.id+" "+s1.name);
//printing members with a white space
//System.out.println("Name is  "+s1.name);
 }
}
```

Objects

- **Object Definitions:**
  - An object is *a real-world entity*.
  - An object is *a runtime entity*.
  - The object is *an entity which has state and behavior*.
  - The object is *an instance of a class*.

Instance variable in Java

- A variable which is created inside the class but outside the method is known as an instance variable.
- Instance variable doesn't get memory at compile time.
- It gets memory at runtime when an object or instance is created.
- That is why it is known as an instance variable.

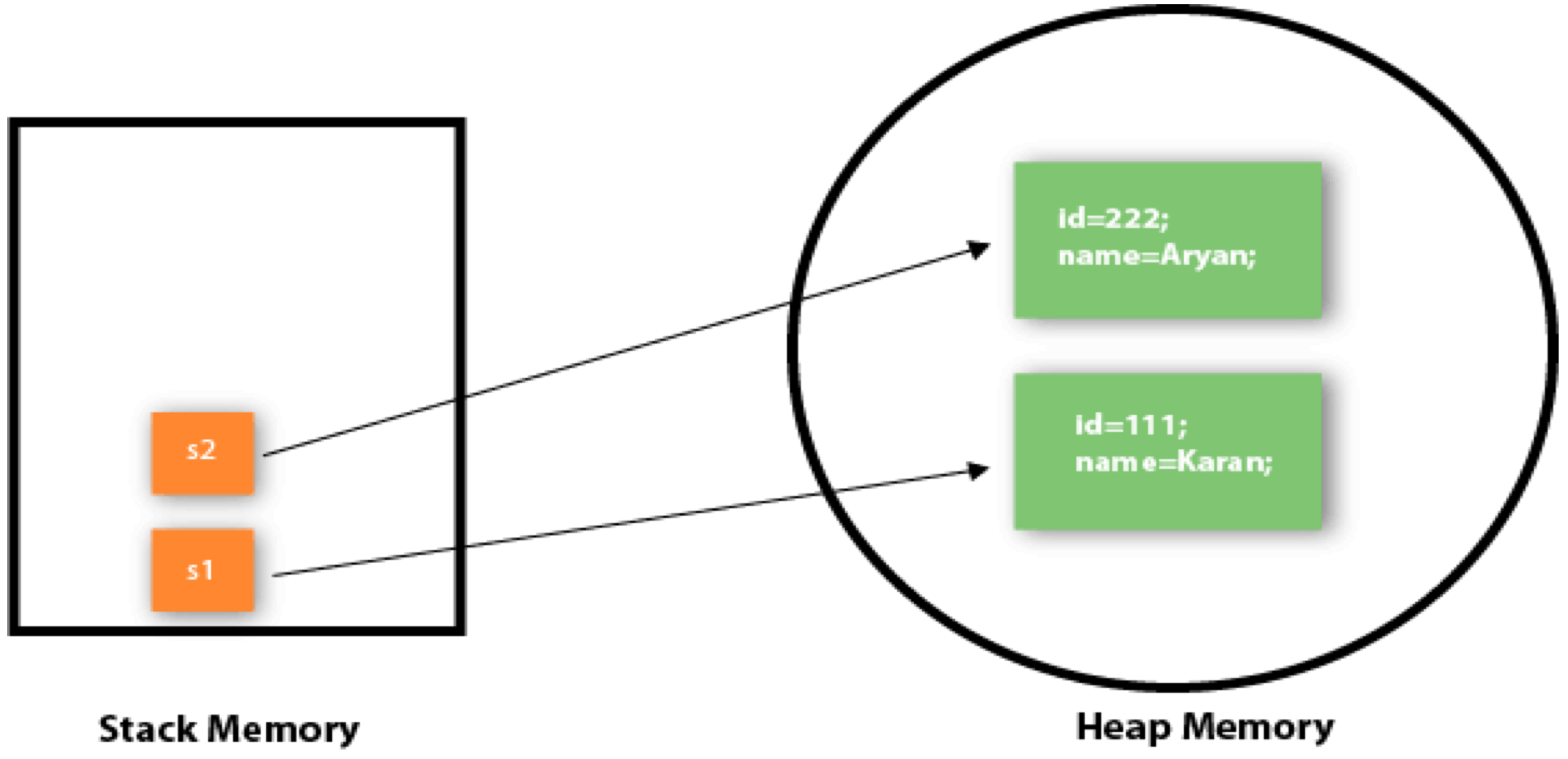
- new keyword in Java
  - The new keyword is used to allocate memory at runtime. All objects get memory in Heap memory area.
- 3 Ways to initialize object
  - By reference variable
  - By method
  - By constructor

```
class TestStudent4
 {
     public static void main(String args[])
     {
        Student s1=new Student();
        Student s2=new Student();
         Student s3=new Student();

        s1.insertRecord(111,"Karan");
        s2.insertRecord(222,"Aryan");

        s1.displayInformation();
        s2.displayInformation();
     }
}
```

Memory representation



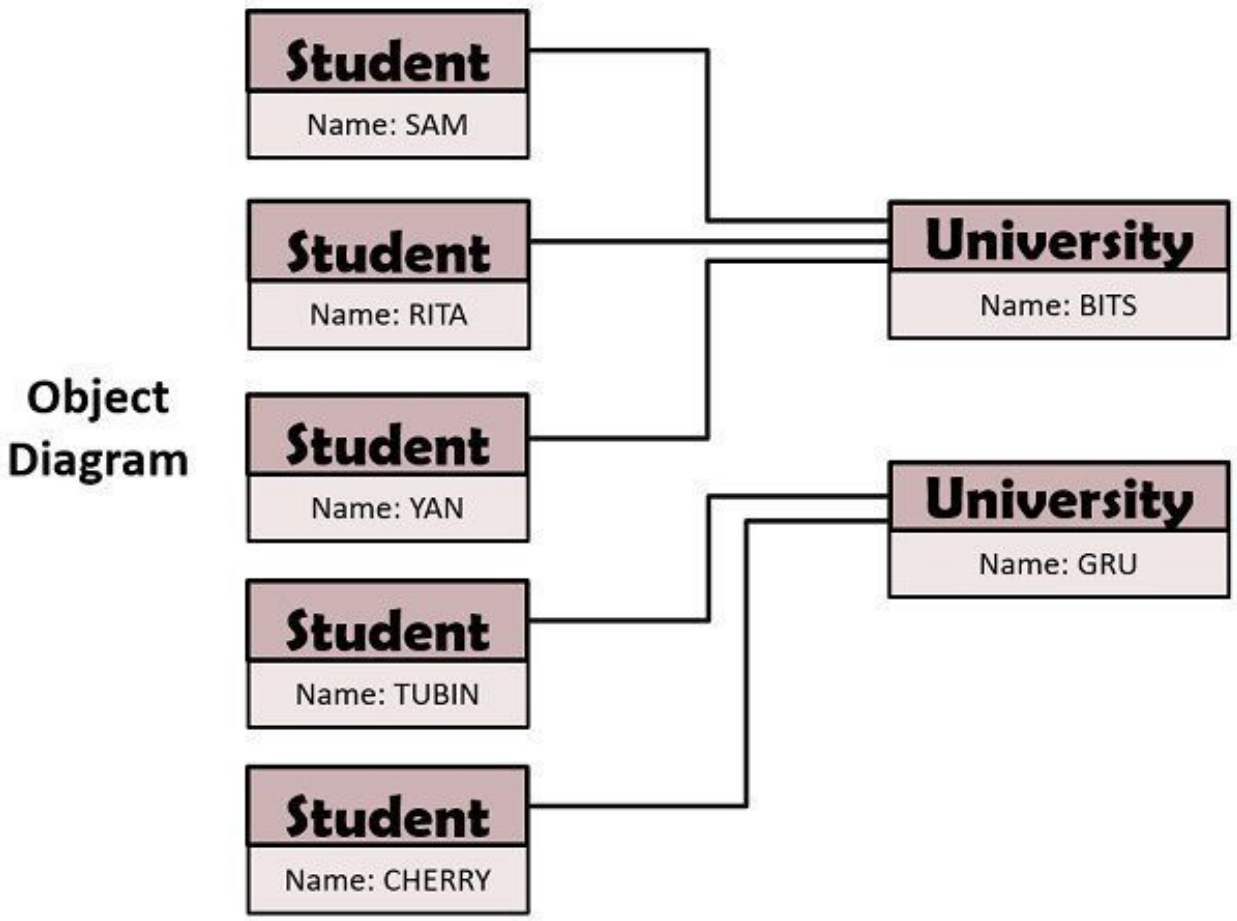
Creating multiple objects by one type only

- **int** a=10, b=20;
- Rectangle r1=**new** Rectangle(), r2=**new** Recta ngle(); //creating two objects

Relationships between the objects

- Links
  - A link is a physical or conceptual connection among objects.
  - Eg. PD *works for* YCCE.
  - Most links relate two objects.
  - UML notation for link is line between objects.
  - In programming links can not be modeled as it is.

Link

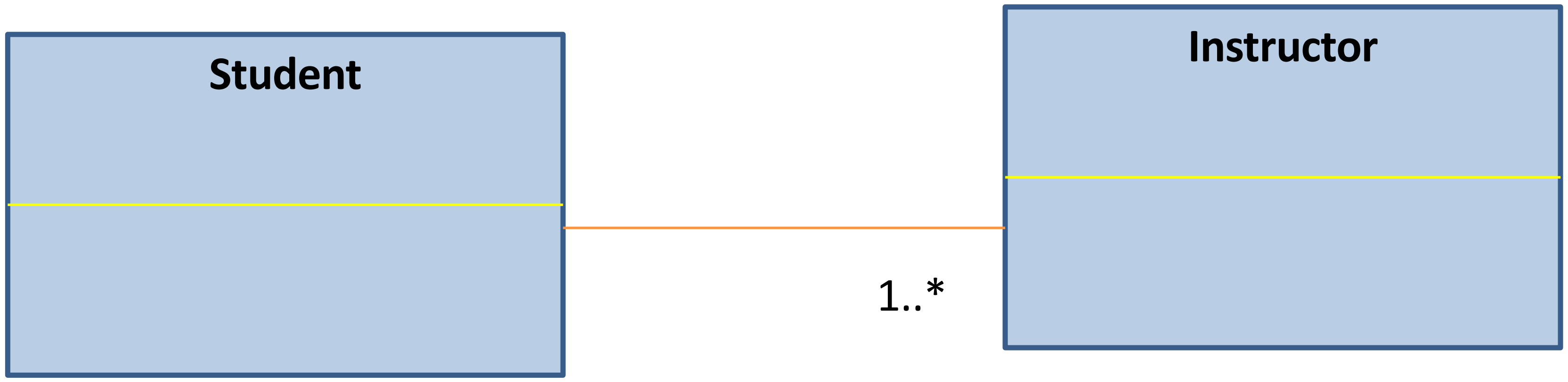

Object Diagram
Student
Name: SAM
Student
Name: RITA
Student
Name: YAN
Student
Name: TUBIN
Student
Name: CHERRY
University
Name: BITS
University
Name: GRU

Association

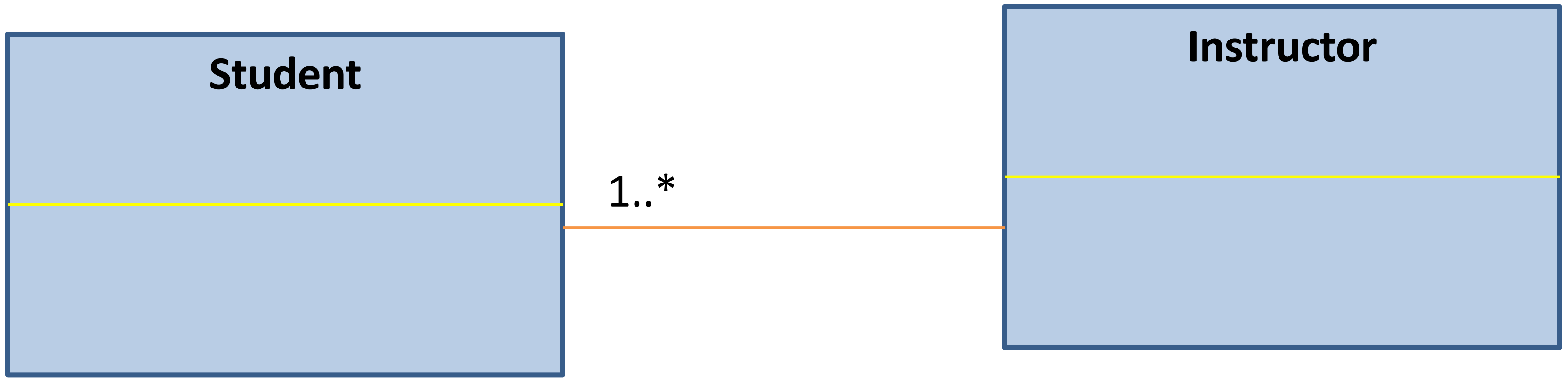
- Description of a group of links.
- Eg. Person ***Works for*** Company.
- In programming association is implemented as reference from one object to another.
- A **reference** is an attribute in one object that refers to another object

- If two classes in a model need to communicate with each other, there must be a link between them, and that can be represented by an association (connector).
- Association can be represented by a line between these classes **with an arrow indicating the navigation direction**
- We can indicate the multiplicity of an association by adding multiplicity adornments to the line denoting the association

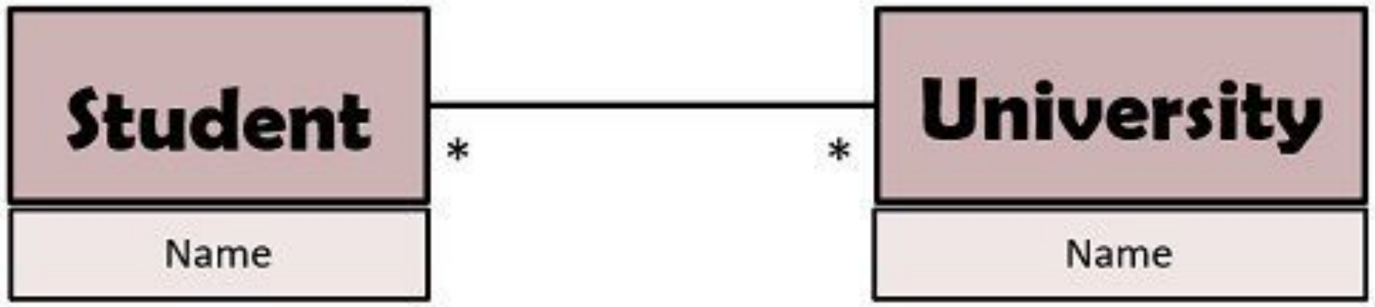
- A single student can associate with multiple teachers:



- The example indicates that every Instructor has one or more Students:



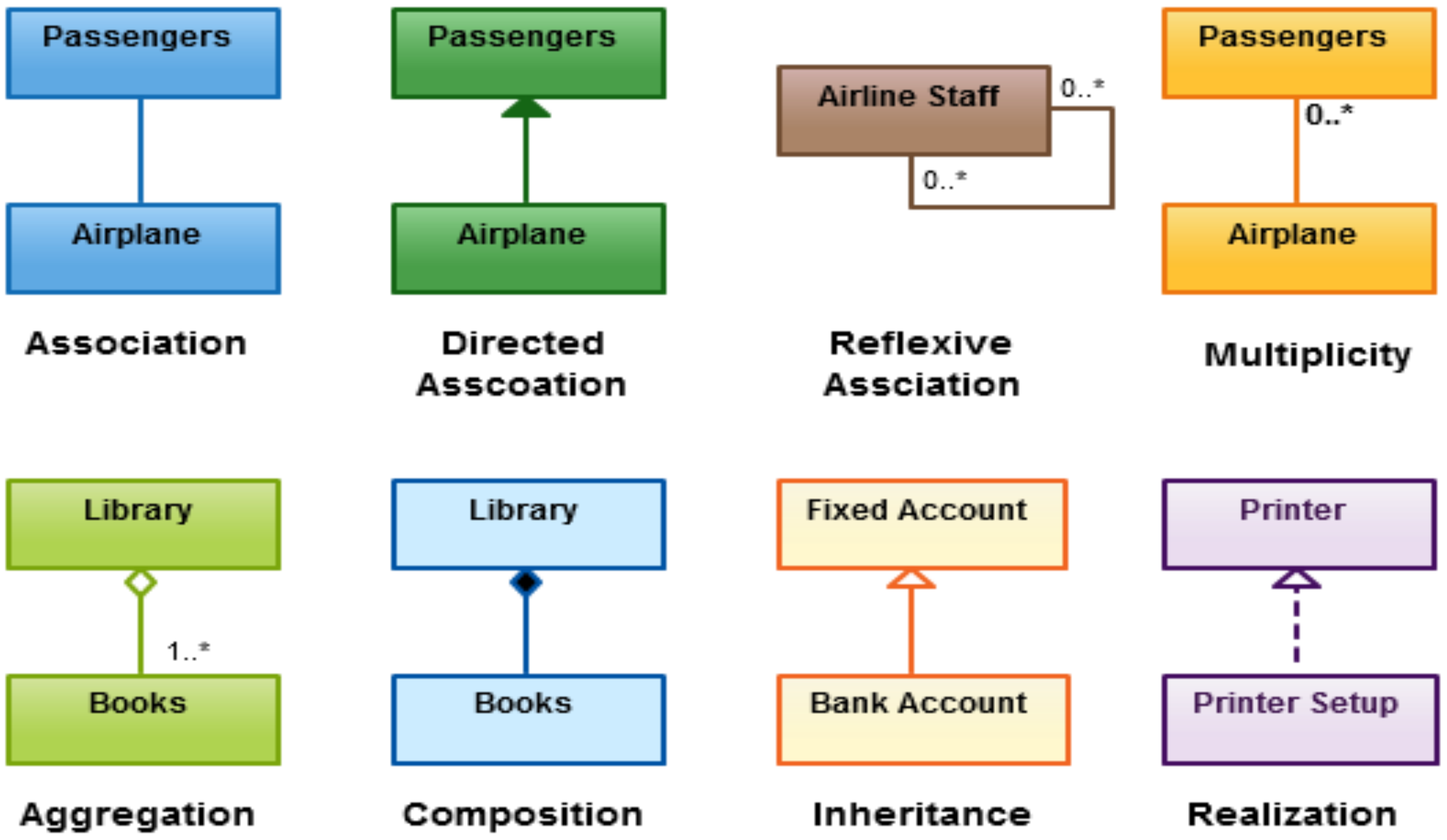
Class Diagram



- **Key Differences Between Link and Association**

1. The logical and physical connection between objects is known as links. On the other hand, a collection of links are specified by an association.
2. The common function of a link is to describe the relationship between objects and connect them with each other. In contrast, an association is used to connect related classes.
3. The UML symbol for link and association are same despite the fact that in the association there is the line segment that shows the relation between two or more classes. As against, in the link, the line segment shows the relationship between objects and group of objects.

UML Notations for Relationships between classes and objects


Passengers
Airplane
Association
Passengers
Airplane
Directed Asscoation
Airline Staff
0..*
0..*
Reflexive Assciation
Passengers
0..*
Airplane
Multiplicity
Library
1..*
Books
Aggregation
Library
Books
Composition
Fixed Account
Bank Account
Inheritance
Printer
Printer Setup
Realization