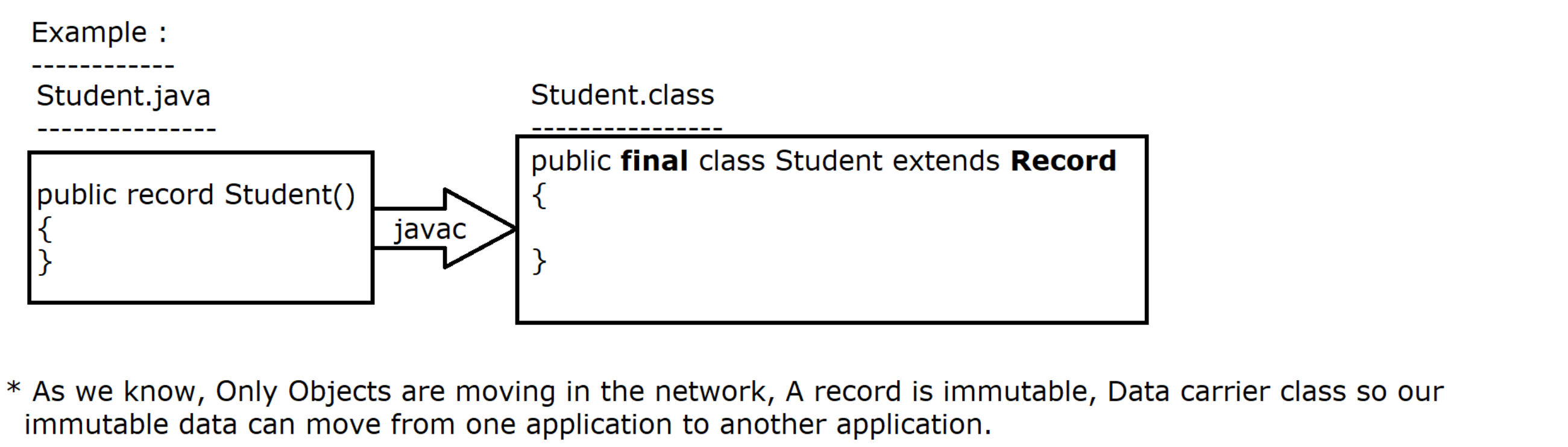
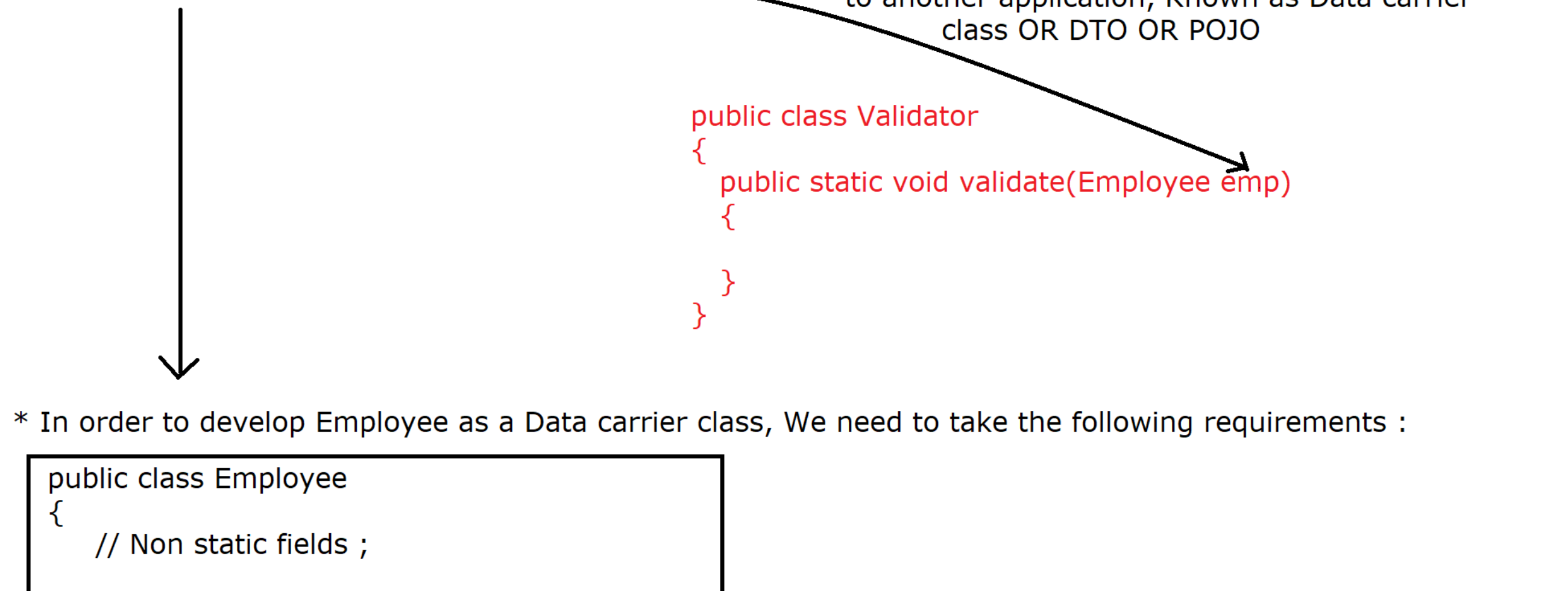


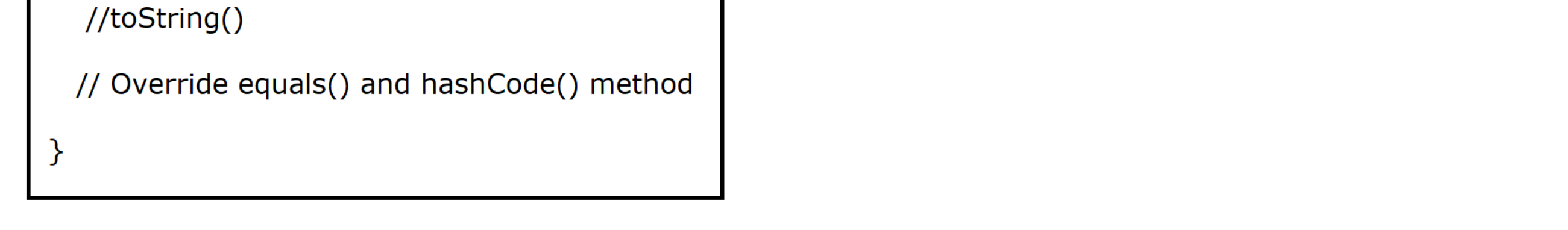
- record in java :
- record is a keyword in java which is used to represent a record and introduced from JDK 17V (Preview version) JDK 14.
- * It is similar to a class because compiler convert this record into a class.
- * record is a keyword whereas we have a predefined class called Record available in java.lang package.



- * As we know, Only Objects are moving in the network, A record is immutable, Data carrier class so our immutable data can move from one application to another application.
- as
- * It is also known DTO (Data transfer Object) OR POJO (Plain Old Java Objects) classes.
- * The main purpose of record to concise the code and remove the boiler-plate code.



- * In order to develop Employee as a Data carrier class, We need to take the following requirements :



- * In record, Automatically a constructor will be generated which is known as "Canonical Constructor" and inside this constructor the variables are known as "Components" and these components are by default **final [Re-assignemnt is not possible]**
- * Automatically the implementation of hashCode(), equals(Object obj) and toString() methods are available inside a record.
- * We can validate Outer world data by using compact constructor.
- * By default every record is implicitly final so a class cannot extend a record, on the other hand, Every record extends from java.lang.Record class so a record cannot extend a class explicitly.
- * A record cannot extend a class but it can implement many interfaces.
- * We cannot define non static variable and non static block, We can accept static variable, static block, static method and non static method.
- * record components are by default final so, setter is not available.
- * Componenets are **final and public so getter facility is available, here the component name will become getter method name.**

```
package com.ravi.record;

import java.util.Objects;

public class EmployeeClass
{
    private Integer empId;
    private String empName;

    public EmployeeClass(Integer empId, String empName)
    {
        super();
        this.empId = empId;
        this.empName = empName;
    }

    public Integer getEmpId() {
        return empId;
    }

    public void setEmpId(Integer empId) {
        this.empId = empId;
    }

    public String getEmpName() {
        return empName;
    }

    public void setEmpName(String empName) {
        this.empName = empName;
    }

    @Override
    public String toString() {
        return "EmployeeClass [empId=" + empId + ", empName=" + empName + "]";
    }

    @Override
    public int hashCode()
    {
        return Objects.hash(empId, empName);
    }

    @Override
    public boolean equals(Object obj)
    {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        EmployeeClass other = (EmployeeClass) obj;
        return Objects.equals(empId, other.empId) && Objects.equals(empName, other.empName);
    }
}

package com.ravi.record;

//Canonical Constructor [Components => final]
public record EmployeeRecord(Integer id, String name)
{
    //Compact Constructor
    public EmployeeRecord
    {
        if(id<=0)
        {
            System.err.println("Invalid Id");
        }
    }
}

package com.ravi.record;

public class ClassAndRecordComparison
{
    public static void main(String[] args)
    {
        EmployeeClass cls1 = new EmployeeClass(111, "Scott");
        System.out.println(cls1);
        EmployeeClass cls2 = new EmployeeClass(111, "Scott");
        System.out.println(cls1.equals(cls2));
        System.out.println(cls1.hashCode()+" : "+cls2.hashCode());
        System.out.println(cls1.getEmpId()+" : "+cls1.getEmpName());

        System.out.println(".....");
        EmployeeRecord rec1 = new EmployeeRecord(-999, "Raj");
        System.out.println(rec1);
        EmployeeRecord rec2 = new EmployeeRecord(999, "Raj");
        System.out.println(rec1.equals(rec2));
        System.out.println(rec1.hashCode()+" : "+rec2.hashCode());
        System.out.println(rec1.id()+" : "+rec1.name());
    }
}
```

Supplier<T> example by using java new feature Record :