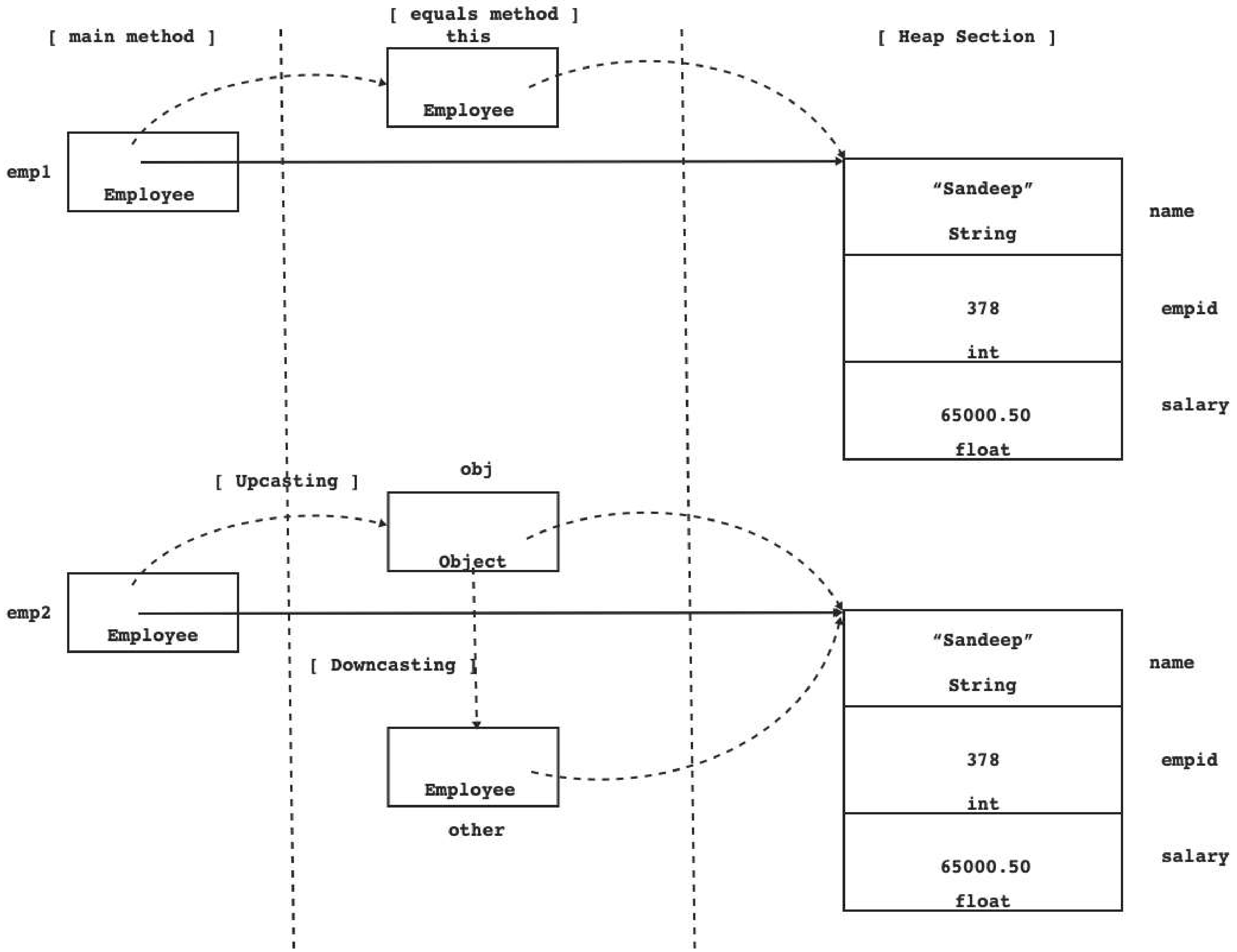
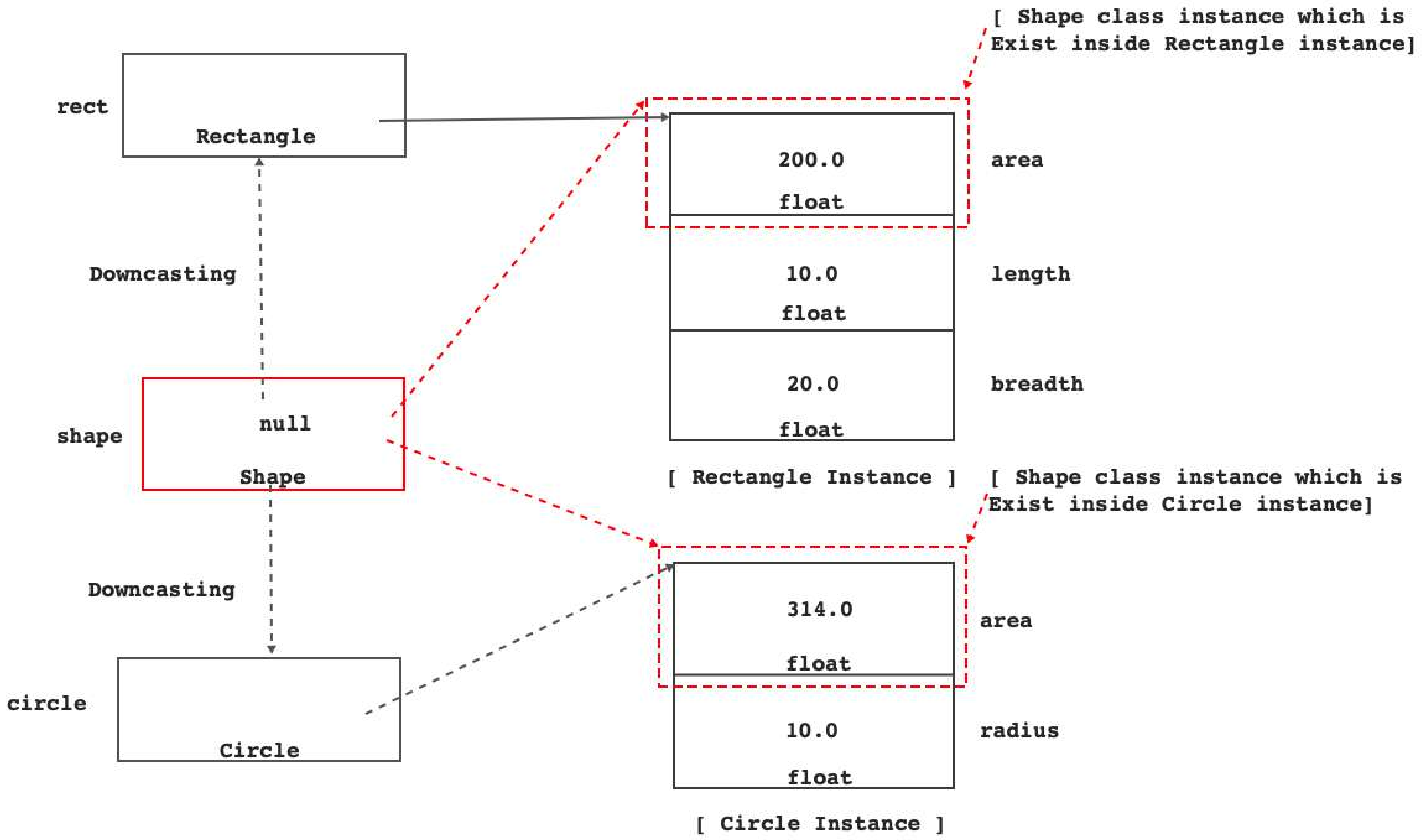
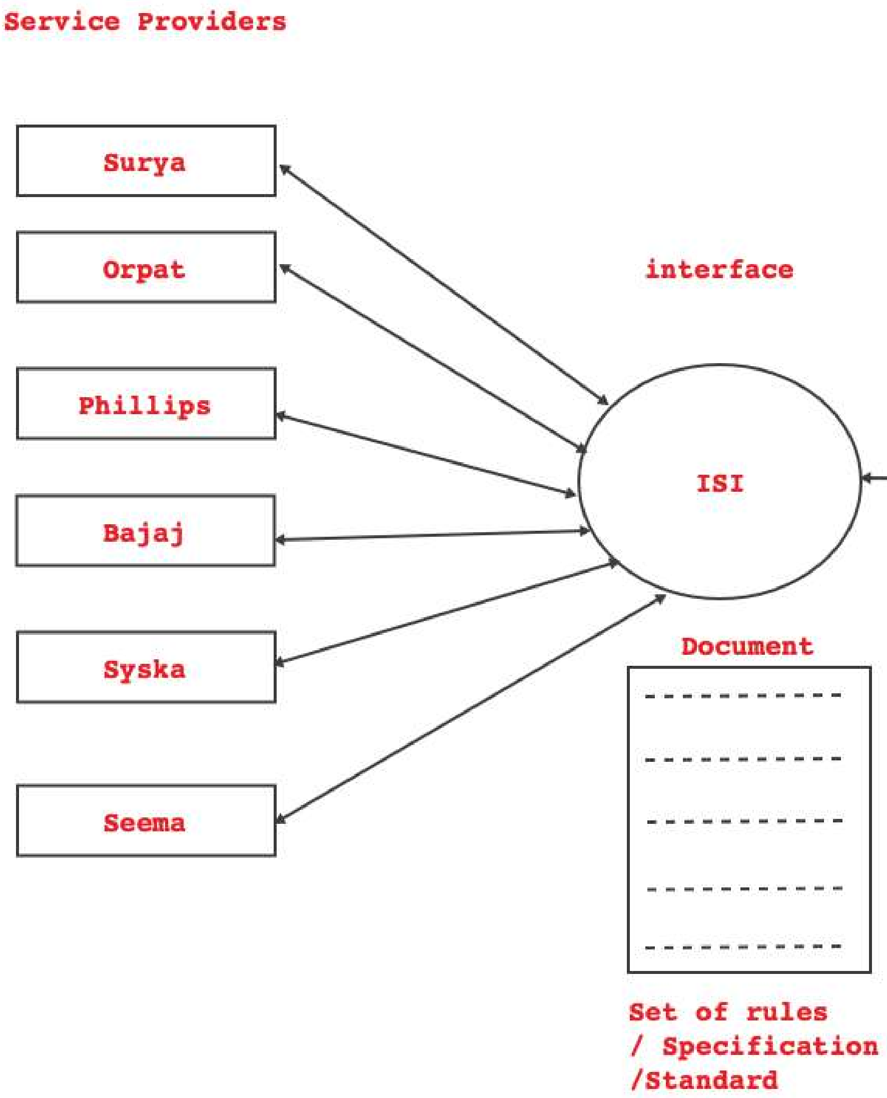

[main method]
[equals method]
this
[Heap Section]
Employee
emp1
Employee
"Sandeep"
String
name
378
int
empid
65000.50
float
salary
obj
[Upcasting]
Object
emp2
Employee
[Downcasting]
"Sandeep"
String
name
378
int
empid
Employee
other
65000.50
float
salary


[Shape class instance which is
Exist inside Rectangle instance]
rect
Rectangle
200.0
float
area
Downcasting
10.0
float
length
20.0
float
breadth
shape
null
Shape
[Rectangle Instance]
[Shape class instance which is
Exist inside Circle instance]
Downcasting
314.0
float
area
circle
Circle
10.0
float
radius
[Circle Instance]



Service Providers

Surya

Orpat

Phillips

Bajaj

Syska

Seema

interface

ISI

Service Consumer

Sandeep

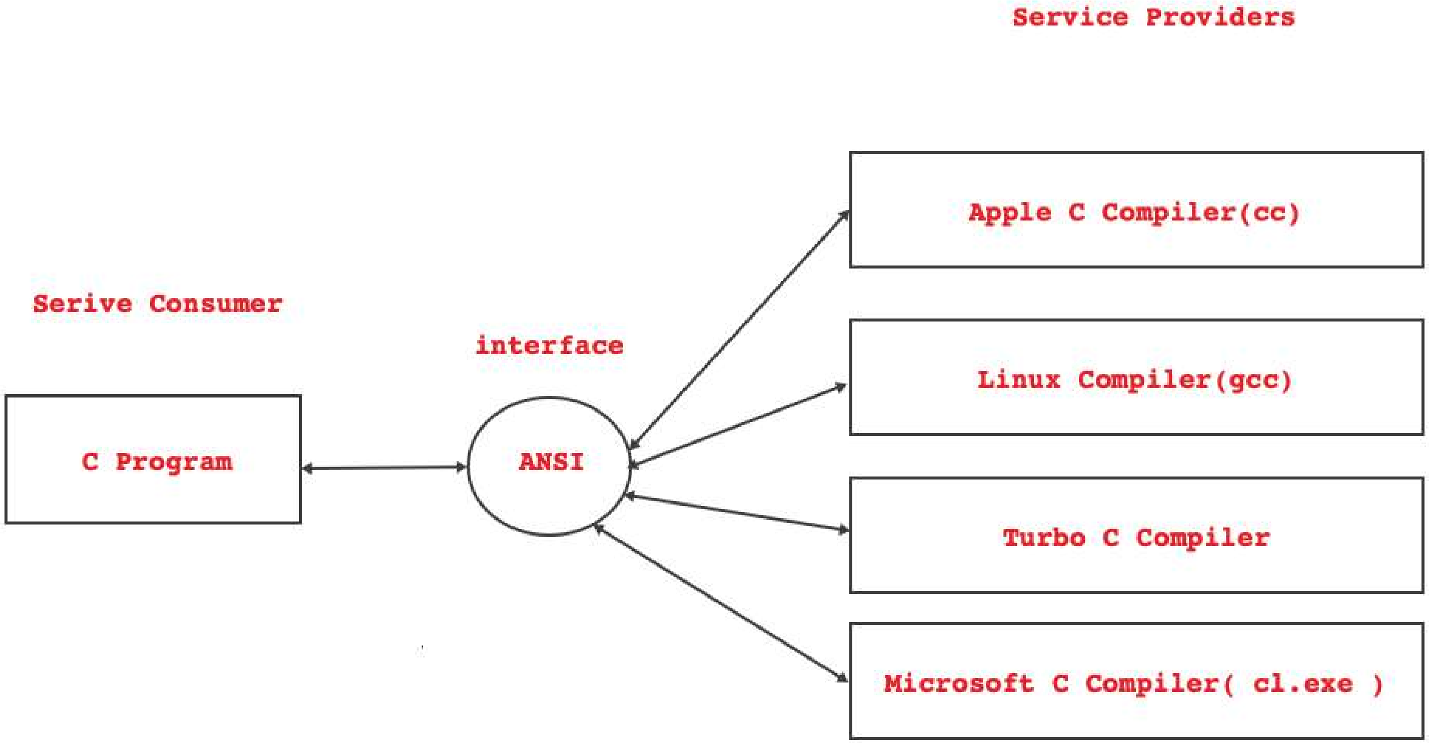
Document

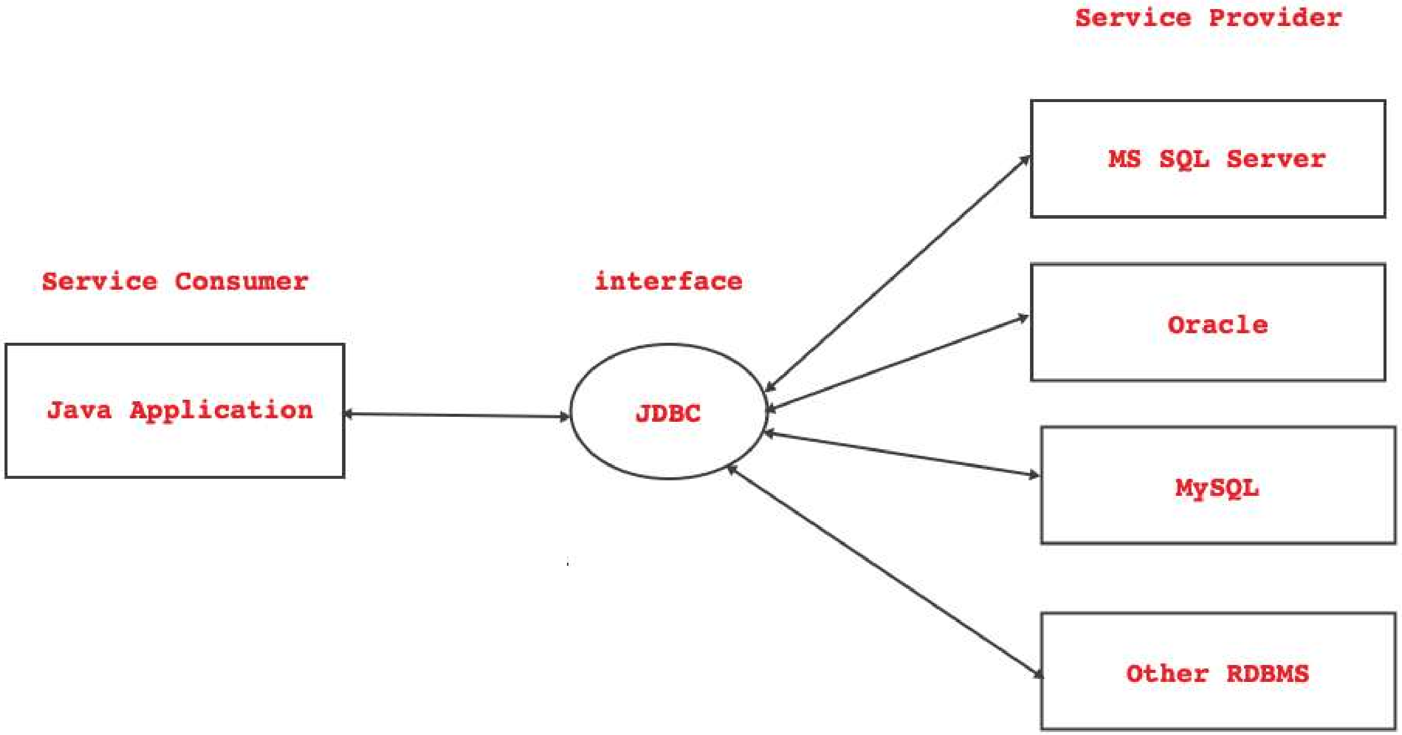
Set of rules
/ Specification
/Standard

1. Power Consumption

2. Warranty

3. Price


Service Providers
Serive Consumer
interface
C Program
ANSI
Apple C Compiler(cc)
Linux Compiler(gcc)
Turbo C Compiler
Microsoft C Compiler(cl.exe)


Service Provider
Service Consumer
interface
MS SQL Server
Oracle
Java Application
JDBC
MySQL
Other RDBMS

- Inheritance
    - parent class - super class
    - child class - sub-class
    - members of super-class inherits into sub-class
    - single , multilevel , hierarchical , hybrid (combination)
    - extends keyword is used

- "super" keyword
    - we want paramterized ctor from sub-class ctor we use "super" keyword
    - super statment is first statement
    - super keyword is used to access methods (non -private) from sub-class
    - if there is shadowing "super" keyword is compulsory
    - if name of super-class method and name of sub-class method is not same you can use "this" or "super" (super keyword is optional)
- Overriding
    - Process of redefining methods from super-class into sub-class with same signature is called overriding

- Rules of method overriding
  - Every method can be overrided unless it private , static , final
  - sub-class access modifier can be same or wider than super-class access modifier
  - super-class return type can be same or sub-class of super-class return type(covariant)
  - Exception handling

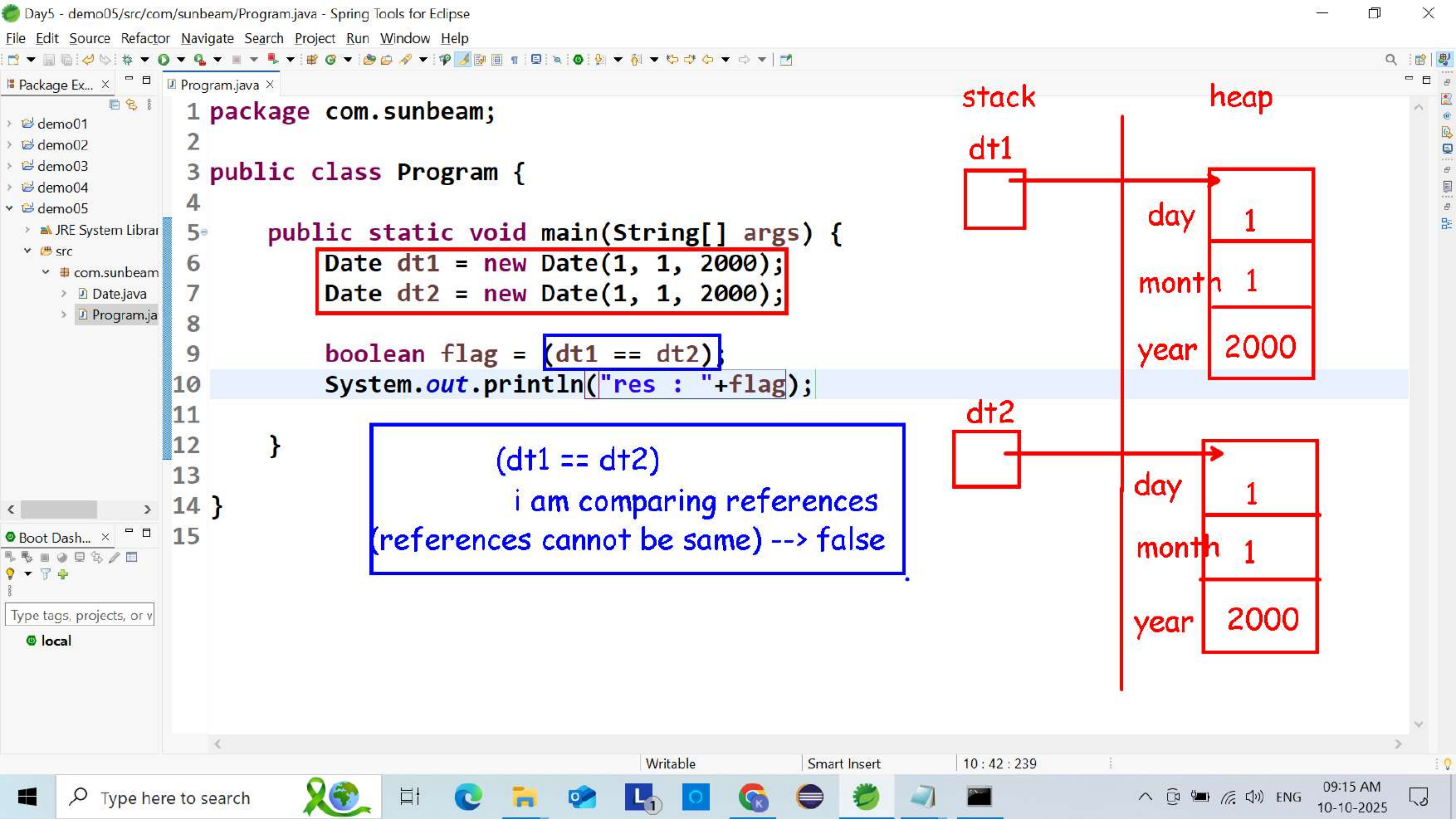
- Upcasting
  - Assigning sub-class reference to super-class reference is called upcasting

    Person p = new Employee(...);

- Downcasting
  - Assigning super-class reference back to sub-class reference is called as downcasting

    Employee emp = (Employee) p; // downcasting

  - If downcasting fails it throws ClassCastException
- Dynamic method dispatch
  - It is called as runtime polymorphism
  - Process of calling sub-class method on super-class reference is called as Dynamic method dispatch (Over-rided methods)
  - Non- overrided methods -- Downcasting |

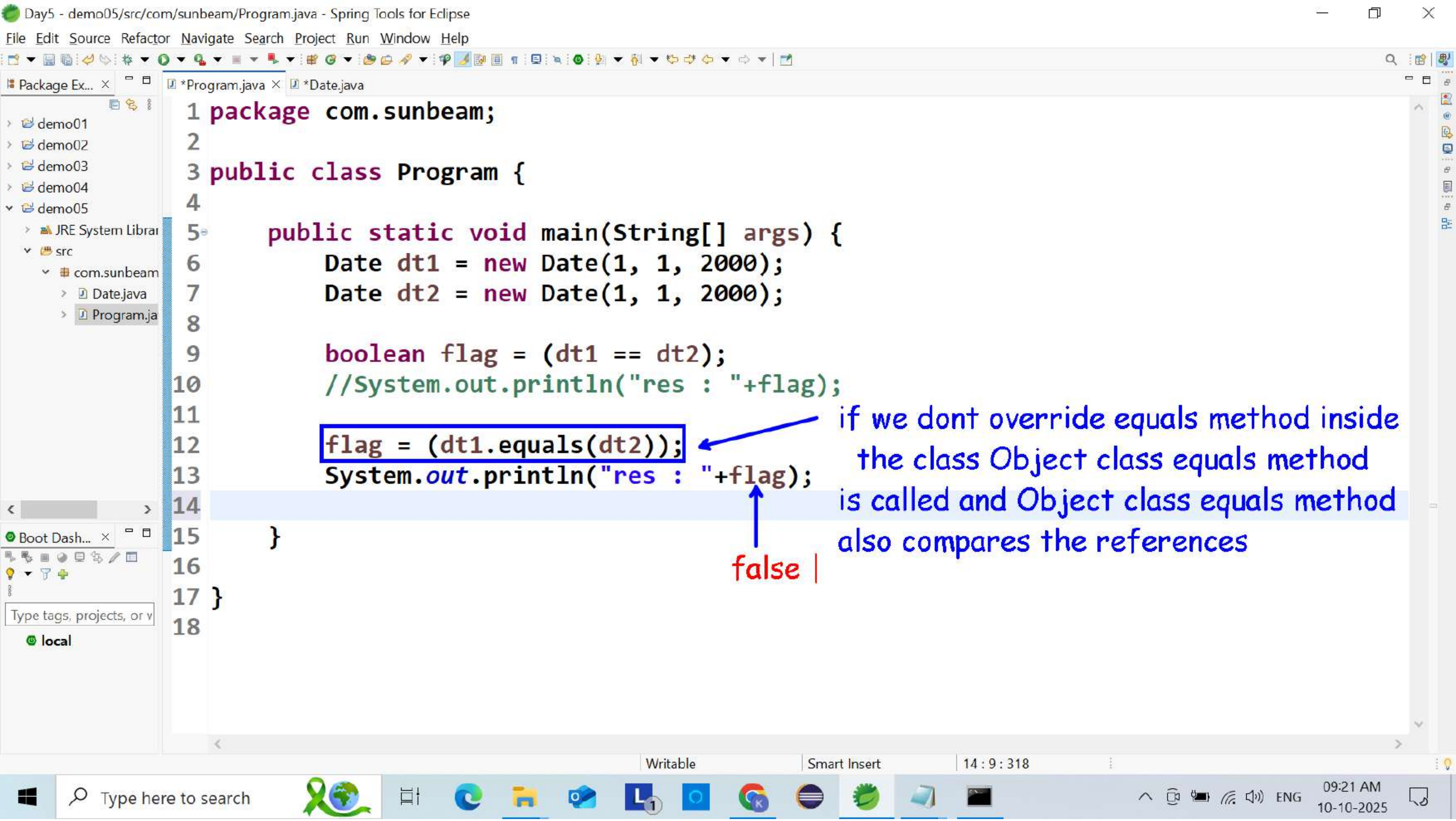


```java
package com.sunbeam;

public class Program {

    public static void main(String[] args) {
        Date dt1 = new Date(1, 1, 2000);
        Date dt2 = new Date(1, 1, 2000);

        boolean flag = (dt1 == dt2);
        System.out.println("res : "+flag);

    }

}
```

(dt1 == dt2)

i am comparing references

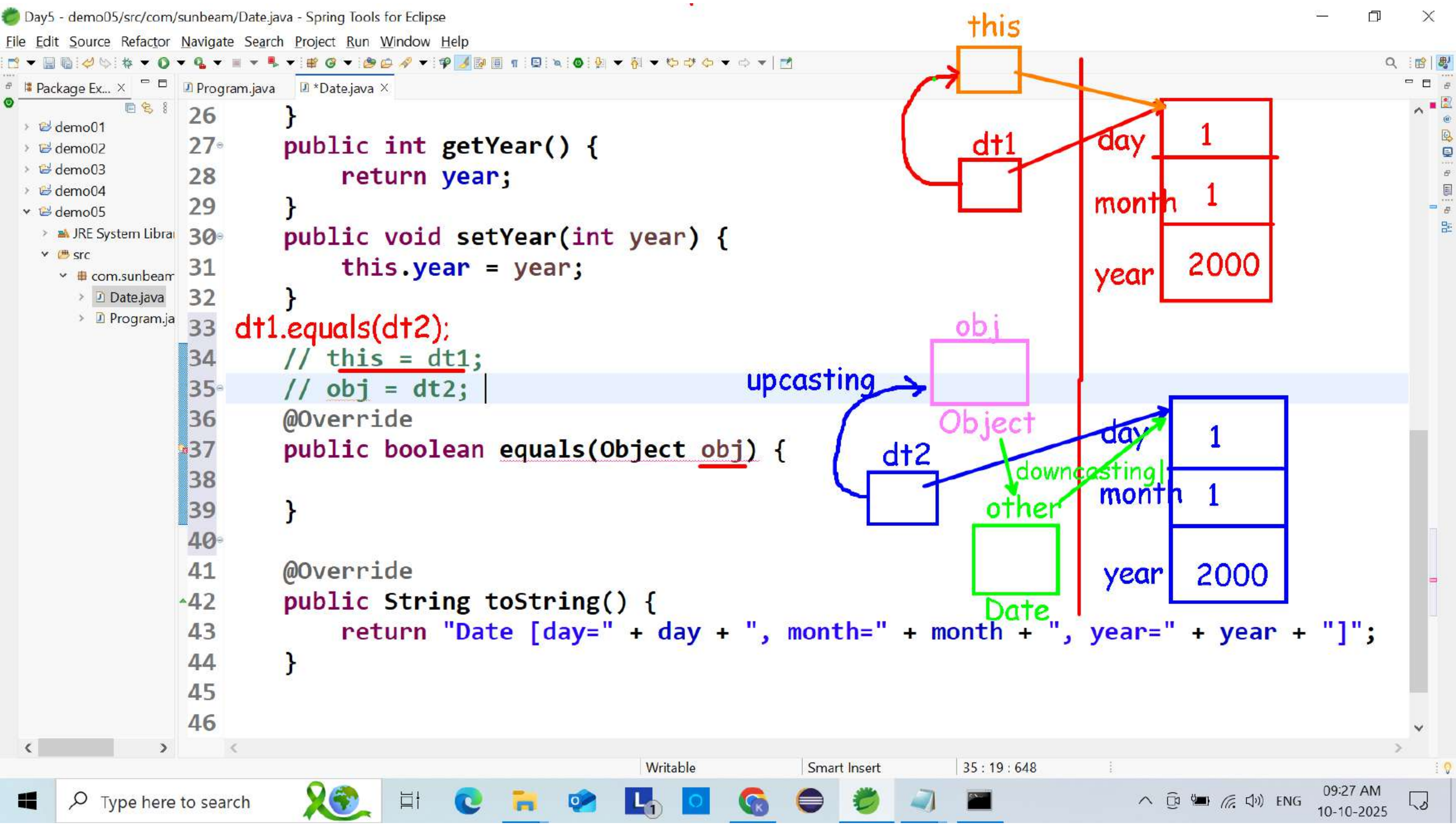
(references cannot be same) --> false



```
package com.sunbeam;

public class Program {

    public static void main(String[] args) {
        Date dt1 = new Date(1, 1, 2000);
        Date dt2 = new Date(1, 1, 2000);

        boolean flag = (dt1 == dt2);
        //System.out.println("res : "+flag);

        flag = (dt1.equals(dt2));
        System.out.println("res : "+flag);

    }

}
```

if we dont override equals method inside the class Object class equals method is called and Object class equals method also compares the references

false



```java
    }
    public int getYear() {
        return year;
    }
    public void setYear(int year) {
        this.year = year;
    }

    // this = dt1;
    // obj = dt2;
    @Override
    public boolean equals(Object obj) {

    }

    @Override
    public String toString() {
        return "Date [day=" + day + ", month=" + month + ", year=" + year + "]";
    }


```

- Abstract method

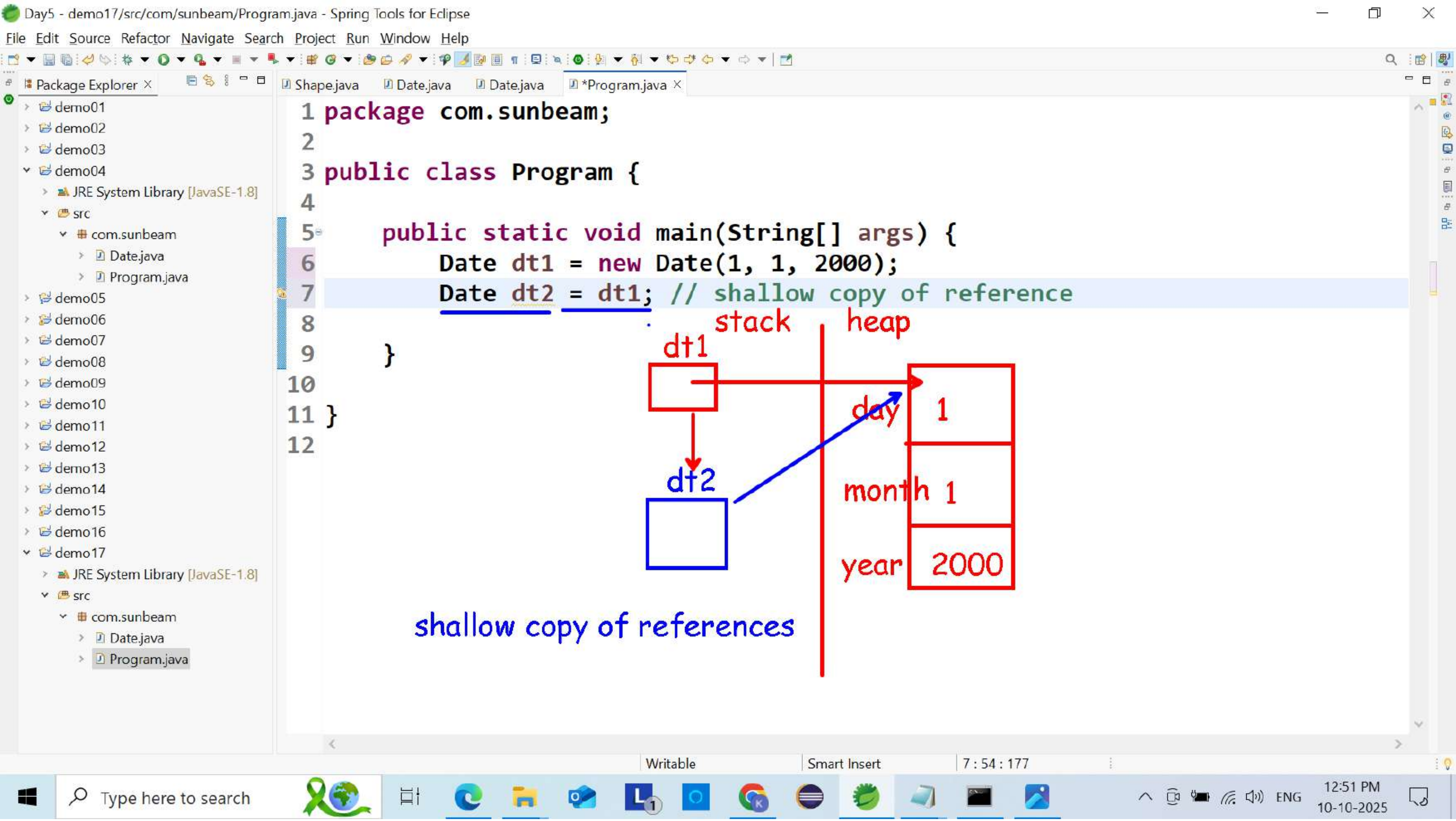
----------------------

- If method is logically 100% incomplete we make method as abstract
- Abstract method do not have body
- IF method is abstract we need to declare class as abstract
- Abstract method cannot private , static , final
- Abstract method need to override inside the subclass otherwise mark subclass as abstract
- Abstract method are forced to be implemented in subclass to have a corresponding behaviour

Abstract class

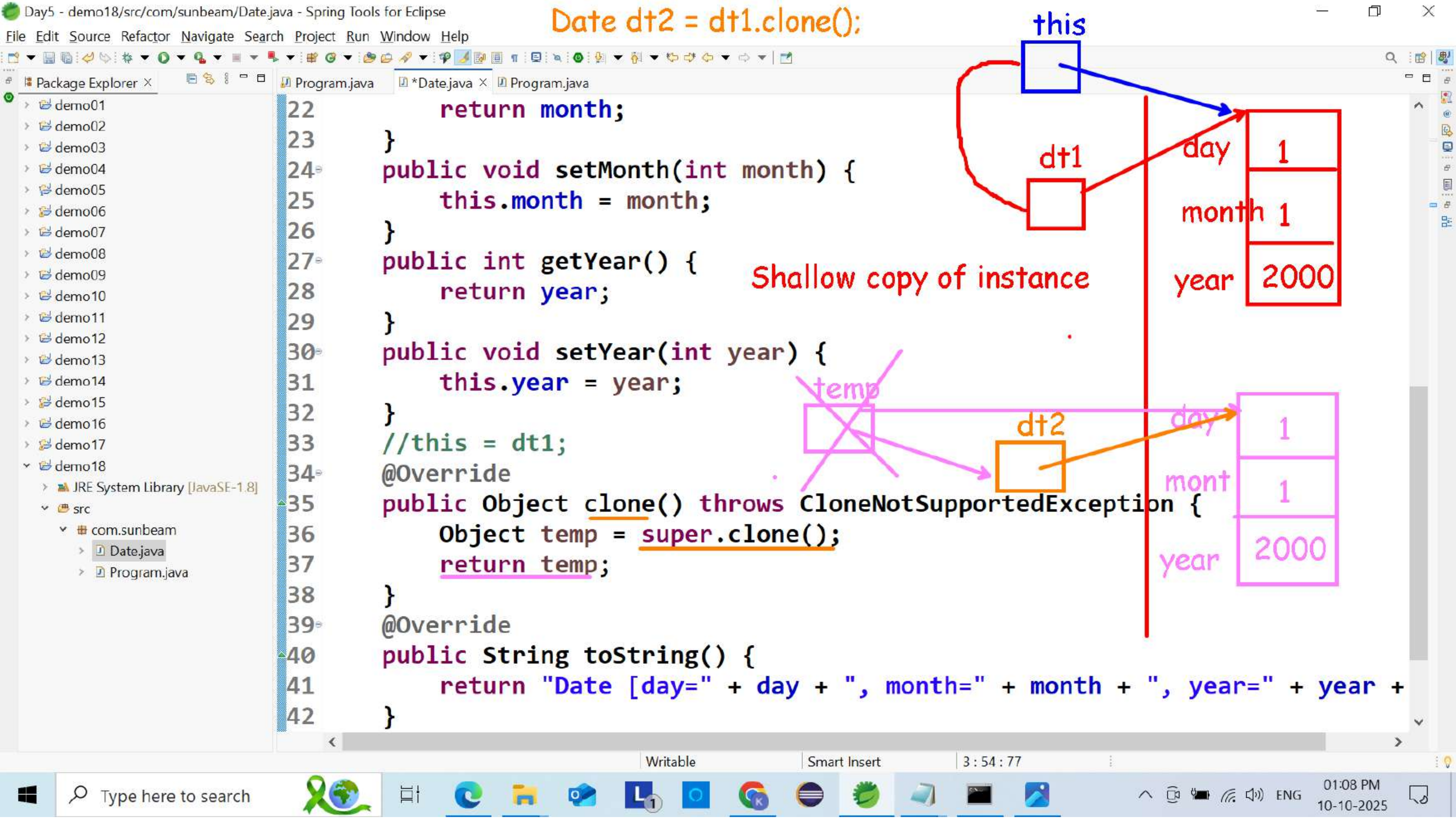
---------

1. If implementation of class is logically Incomplete we declare class as abstract
2.Abstract class can contain zero or more abstract methods
3.Abstract class can have 1 or more abstract methods, ifany method is abstract class shoud be decln as abstract
4.Abstract class can have fields , methods , constructors
5.We can create on reference of abstract class we cannot instantiate abstract class



```java
package com.sunbeam;

public class Program {

    public static void main(String[] args) {
        Date dt1 = new Date(1, 1, 2000);
        Date dt2 = dt1; // shallow copy of reference

    }

}
```

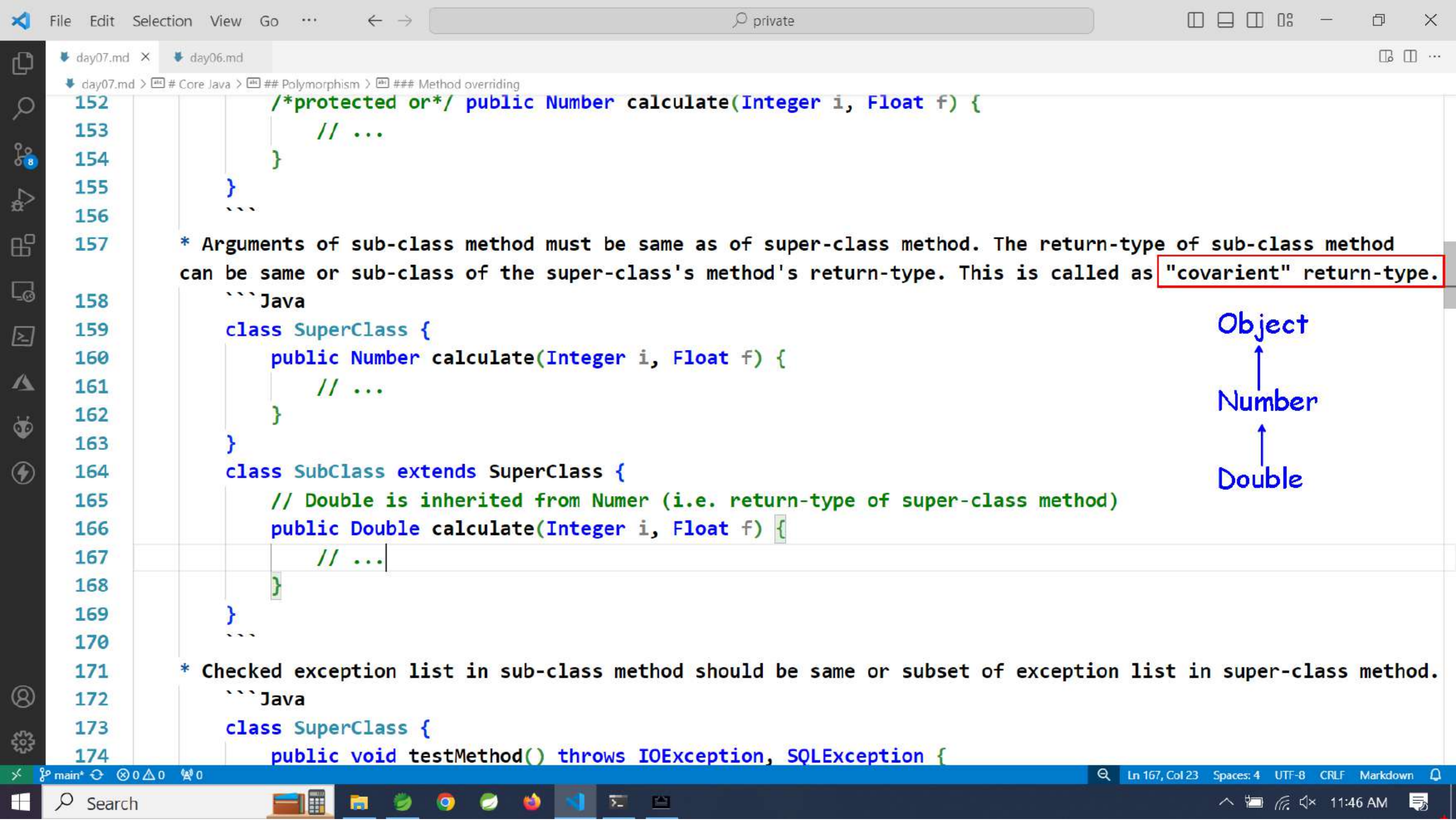
shallow copy of references



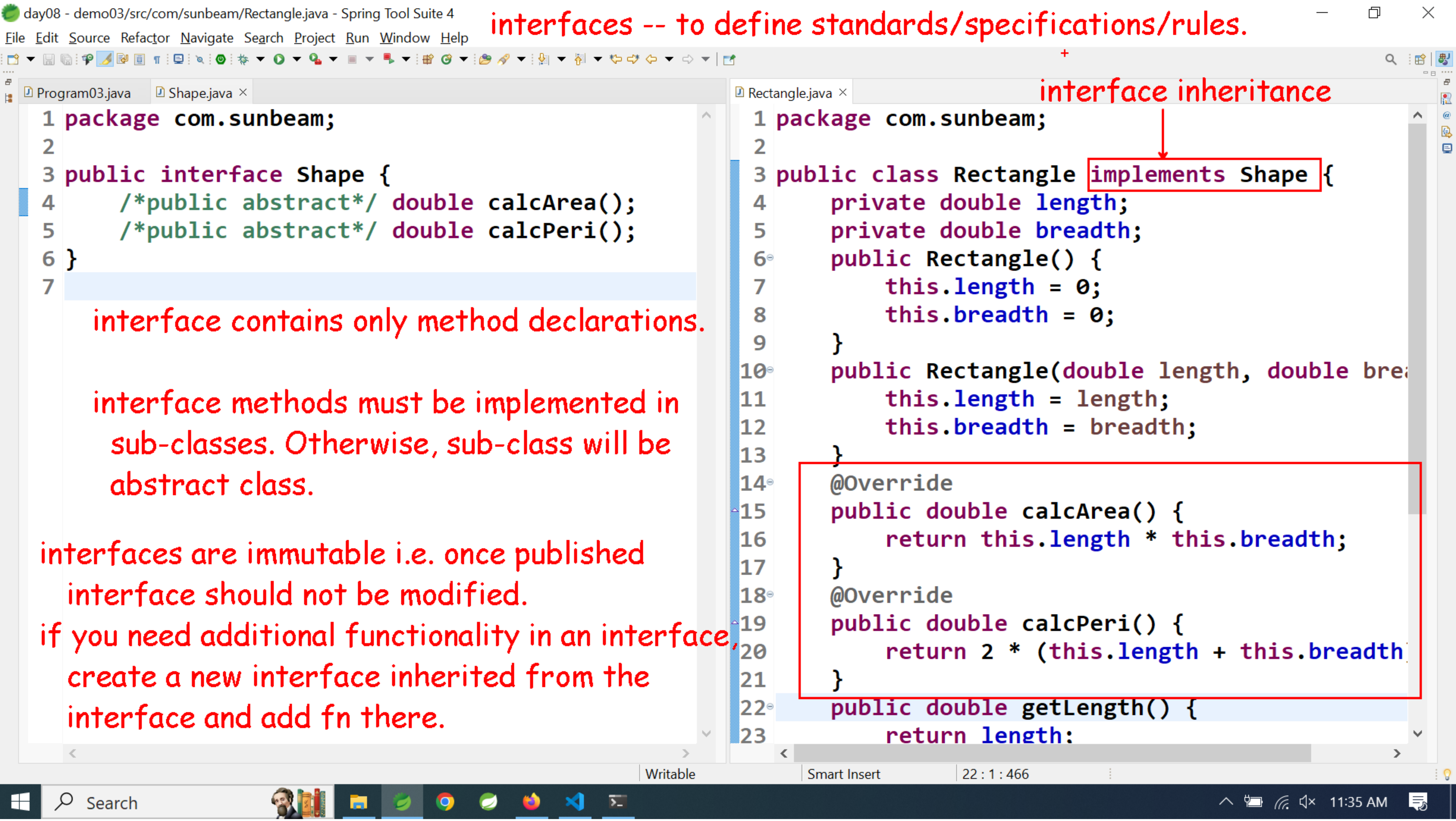
```
Date dt2 = dt1.clone();
```

```
        return month;
    }
    public void setMonth(int month) {
        this.month = month;
    }
    public int getYear() {
        return year;
    }
    public void setYear(int year) {
        this.year = year;
    }
    //this = dt1;
    @Override
    public Object clone() throws CloneNotSupportedException {
        Object temp = super.clone();
        return temp;
    }
    @Override
    public String toString() {
        return "Date [day=" + day + ", month=" + month + ", year=" + year +
    }
```

Shallow copy of instance



```
        /*protected or*/ public Number calculate(Integer i, Float f) {
            // ...
        }
    }
    ```
* Arguments of sub-class method must be same as of super-class method. The return-type of sub-class method can be same or sub-class of the super-class's method's return-type. This is called as "covarient" return-type.
    ```Java
    class SuperClass {
        public Number calculate(Integer i, Float f) {
            // ...
        }
    }
    class SubClass extends SuperClass {
        // Double is inherited from Numer (i.e. return-type of super-class method)
        public Double calculate(Integer i, Float f) {
            // ...
        }
    }
    ```
* Checked exception list in sub-class method should be same or subset of exception list in super-class method.
    ```Java
    class SuperClass {
        public void testMethod() throws IOException, SQLException {
```


day08 - demo03/src/com/sunbeam/Rectangle.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
Program03.java
Shape.java
Rectangle.java
Writable
Smart Insert
22 : 1 : 466
Search
11:35 AM

interfaces -- to define standards/specifications/rules.

```java
package com.sunbeam;

public interface Shape {
    /*public abstract*/ double calcArea();
    /*public abstract*/ double calcPeri();
}

```

interface contains only method declarations.

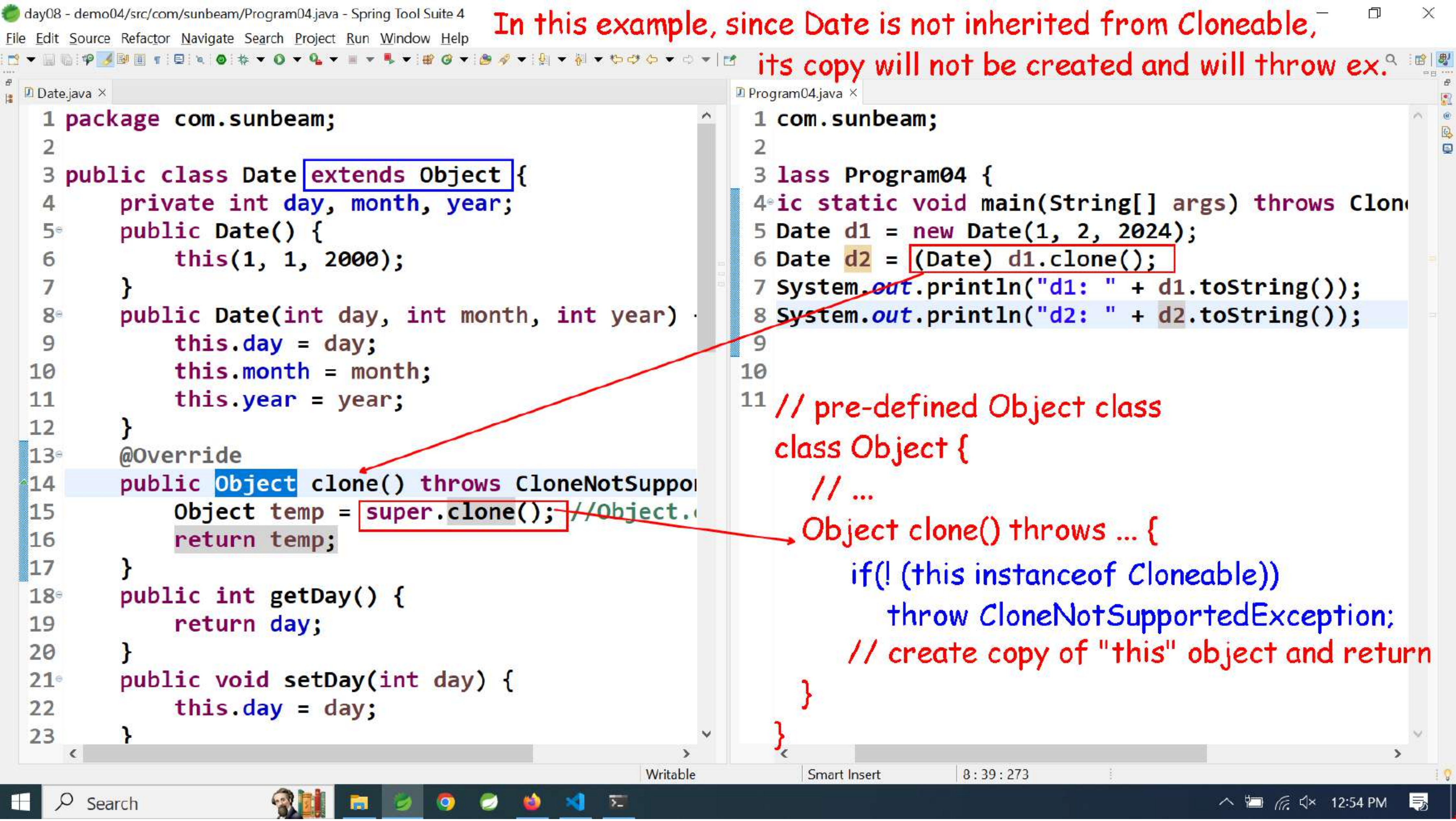
interface methods must be implemented in sub-classes. Otherwise, sub-class will be abstract class.

interfaces are immutable i.e. once published interface should not be modified.

if you need additional functionality in an interface, create a new interface inherited from the interface and add fn there.

interface inheritance

```java
package com.sunbeam;

public class Rectangle implements Shape {
    private double length;
    private double breadth;
    public Rectangle() {
        this.length = 0;
        this.breadth = 0;
    }
    public Rectangle(double length, double brea
        this.length = length;
        this.breadth = breadth;
    }
    @Override
    public double calcArea() {
        return this.length * this.breadth;
    }
    @Override
    public double calcPeri() {
        return 2 * (this.length + this.breadth
    }
    public double getLength() {
        return length;
```


day08 - demo04/src/com/sunbeam/Program04.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
Date.java
Program04.java
Writable
Smart Insert
8 : 39 : 273
Search
12:54 PM

In this example, since Date is not inherited from Cloneable, its copy will not be created and will throw ex.

```java
package com.sunbeam;

public class Date extends Object {
    private int day, month, year;
    public Date() {
        this(1, 1, 2000);
    }
    public Date(int day, int month, int year) ·
        this.day = day;
        this.month = month;
        this.year = year;
    }
    @Override
    public Object clone() throws CloneNotSuppor
        Object temp = super.clone(); //Object.
        return temp;
    }
    public int getDay() {
        return day;
    }
    public void setDay(int day) {
        this.day = day;
    }
```

```java
com.sunbeam;

lass Program04 {
ic static void main(String[] args) throws Clon
Date d1 = new Date(1, 2, 2024);
Date d2 = (Date) d1.clone();
System.out.println("d1: " + d1.toString());
System.out.println("d2: " + d2.toString());
```

```java
// pre-defined Object class
class Object {
   // ...
   Object clone() throws ... {
       if(! (this instanceof Cloneable))
          throw CloneNotSupportedException;
       // create copy of "this" object and return
   }
}
```