Method Overriding — Day 5.

- Rules to Override in Java VVVImp

① private, static, final method cannot be overridden.

देख Rule ② subclass method must have same or wider access modifier than super-class method.

private (lowest)
default
protected
public (highest)

③ Arguments of subclass method must be same as super class method.

देख Rule ④ The return type of sub-class method can be same of super class → or sub class of the super-class's method's ... method return type. This is called "covariant" return type.

⑤ Checked exception list in sub-class method should be same or subset of exception list in super class method.

@Override annotation on (subclass method) ⇒ checks if we follows the Rules of overriding (Java 5.0)

overriding
① upcasting required
② base class pointer → derived class & method call करता.

Run Time Polymorphism — (demo03)

- Calling sub class method on superclass reference is called Run Time Polymorphism

Shape.calculateArea();

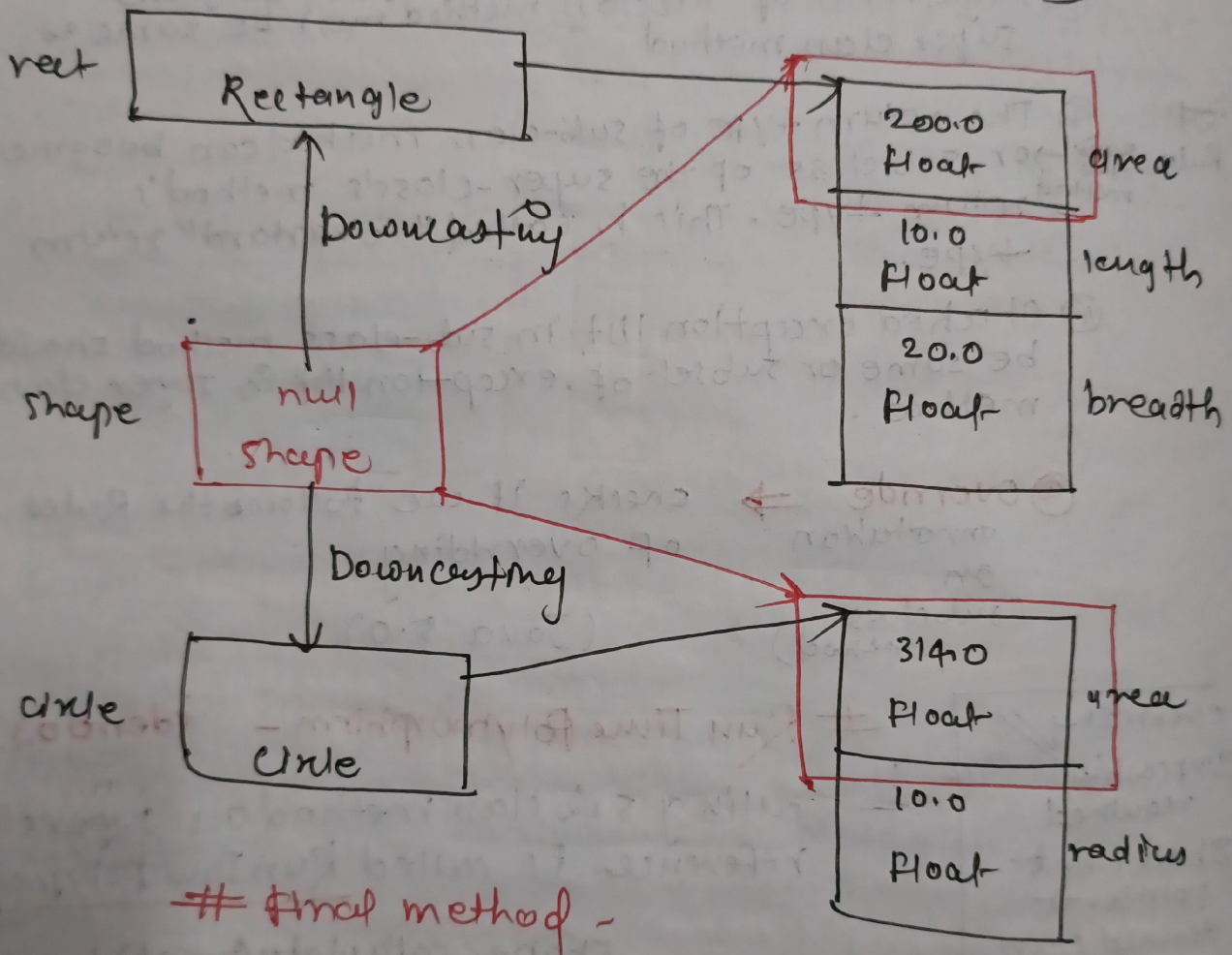
Here we achive RTP in (demo03).

- The method to be called is decided at runtime depending on type of object
- Late Binding or Dynamic Method Dispatch.

Downcasting :

- To call the non overrided function.
- we cant use Superclass reference for that, we have to use subclass reference
  Therefore we need downcasting.

Instanceof (In place of typeid)



Final method -

- If implementation of a superclass method is logically complete, then we declare method final.
- Cannot be overridden
- Can be inherited into subclass.

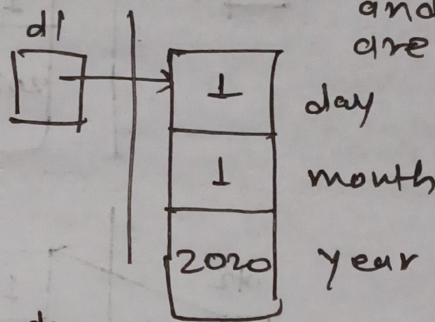
final class -

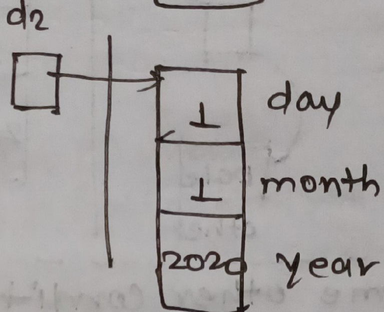
- If implementation of a class is 100% complete then final
- Cannot be inherited
- All wrapper classes are final

.equals method implementation

- To compare the state of object

① Date d1 = new Date(1, 1, 2020);
Date d2 = new Date(1, 1, 2020);
boolean flag = (d1 == d2); ⇒ Here we are comparing reference and references are different
sout(flag) ⇒ false





② boolean flag (d1.equals(d2)) ⇒ Here we don't override .equals method.
sout (flag) ⇒ false

And ... If we don't override the equals method then object class equals method will be called and object class equals method will also compare the references and references cannot be same.

d1.equals(d2);
आपण d1 वर equals method call केली म्हणजे d1 चा reference जाणार this मध्ये
आणी d2 as a parameter पाठवला आणी downcast केले तर other मध्ये d2 जाणार
// this = d1
// obj = d2

③ @override
public boolean equals(Object obj) { (super class → its object)
Date other = (Date) obj; // downcasting ②
if (this.day == other.day && this.month == other.month && this.year == other.year)
{ return true }
return false;
}

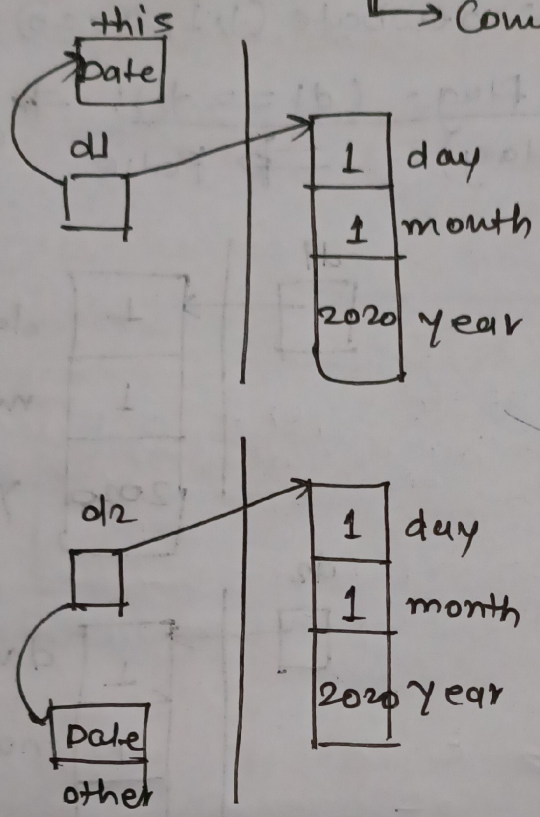
① d2 object आपण as a parameter पाठवत आहे in equals method म्हणजे
Object obj = d2;
// Upcasting

Now we override the equals method -

- By overriding this equals method it will now compare state of object
- Now if we call

```
boolean flag = (d1.equals(d2)) // True
```

↳ Comparing state of object.



Some other conditions.

Topics Revised:
1) Upcasting
2) down casting
3) Instance of
4) this reference

① d1.equals(null);

```
if (obj==null)
return false;
```

② d1.equals(d1);

```
if(this==obj)
return true;
```

// to prevent downcasting when d1=d1

③ Incompatible type (If we are passing String)

```
if (!(obj instance of Date))
return false;
```

Day 1

demo 01 - upcasting & downcasting
demo 02 - Rule 2 of overriding, @override
demo 03 - Run Time Polymorphism.
Shape.calculateArea();
Shape.accept(Record);
demo 04 - InstanceOf (shape example)
Rectangle, circle

downcast - to call non-overrided methods of subclass, methods calling.

demo 05 - explained the working of ~~casting~~ methods calling.
// Important रट लो यार.

demo 06 - final method.
demo 07 - final class
demo 08 - .equals method.
demo 09 - Interface
demo 10 - multiple interface inheritance
demo 11 - diamond problem field problem still exists in Java
demo 12 - method ~~के लिए~~ ambiguity ~~नहीं आती~~.
demo 13 - adapter class Test

demo 14 - How we have seen, when to declare how to declare final, abstract

demo 15 - interface Shape
then class Square, Circle, Rectangle by implementing Shape.