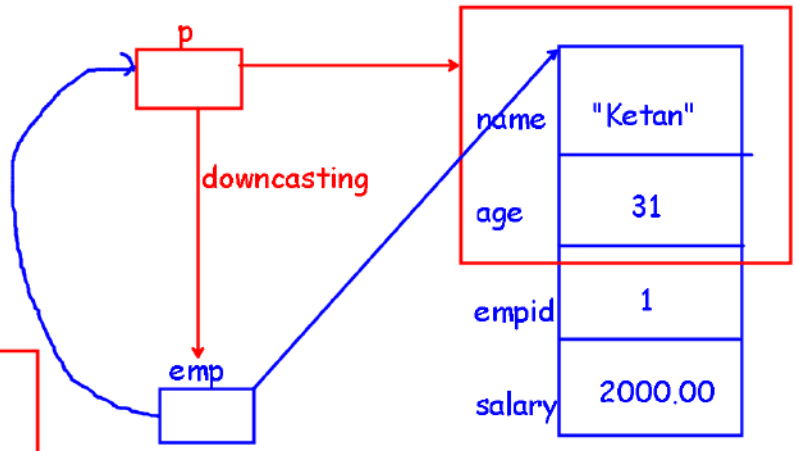
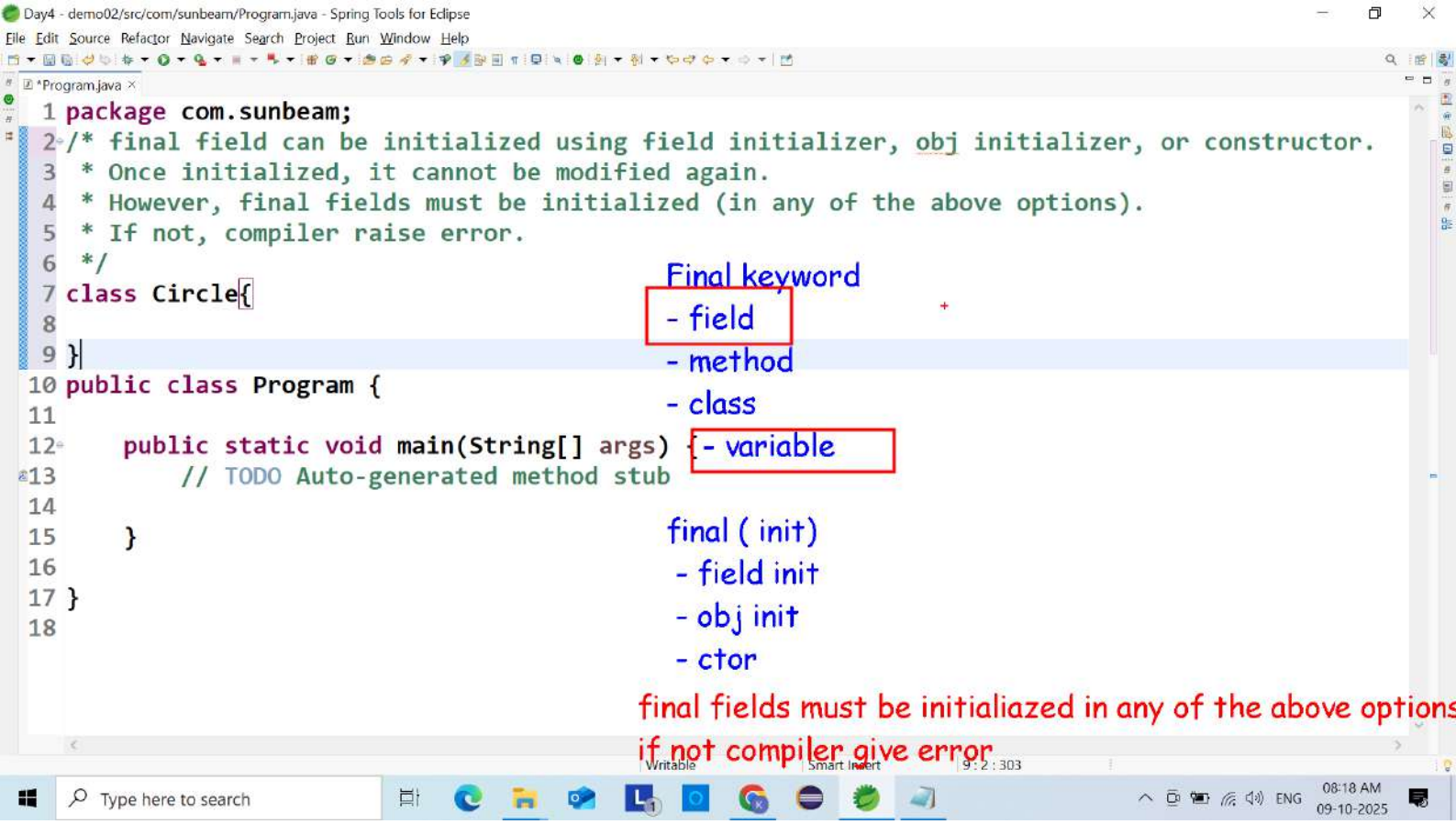
```
Employee emp = new Employee("Ketan",31,1,2000.00);

sysout(emp.name); // OK
sysout(emp.age); // OK
sysout(emp.empid); // OK
sysout(emp.salary); //OK

                    upcasting
Person p = (Person) emp;
sysout(p.name);
sysout(p.age);
```

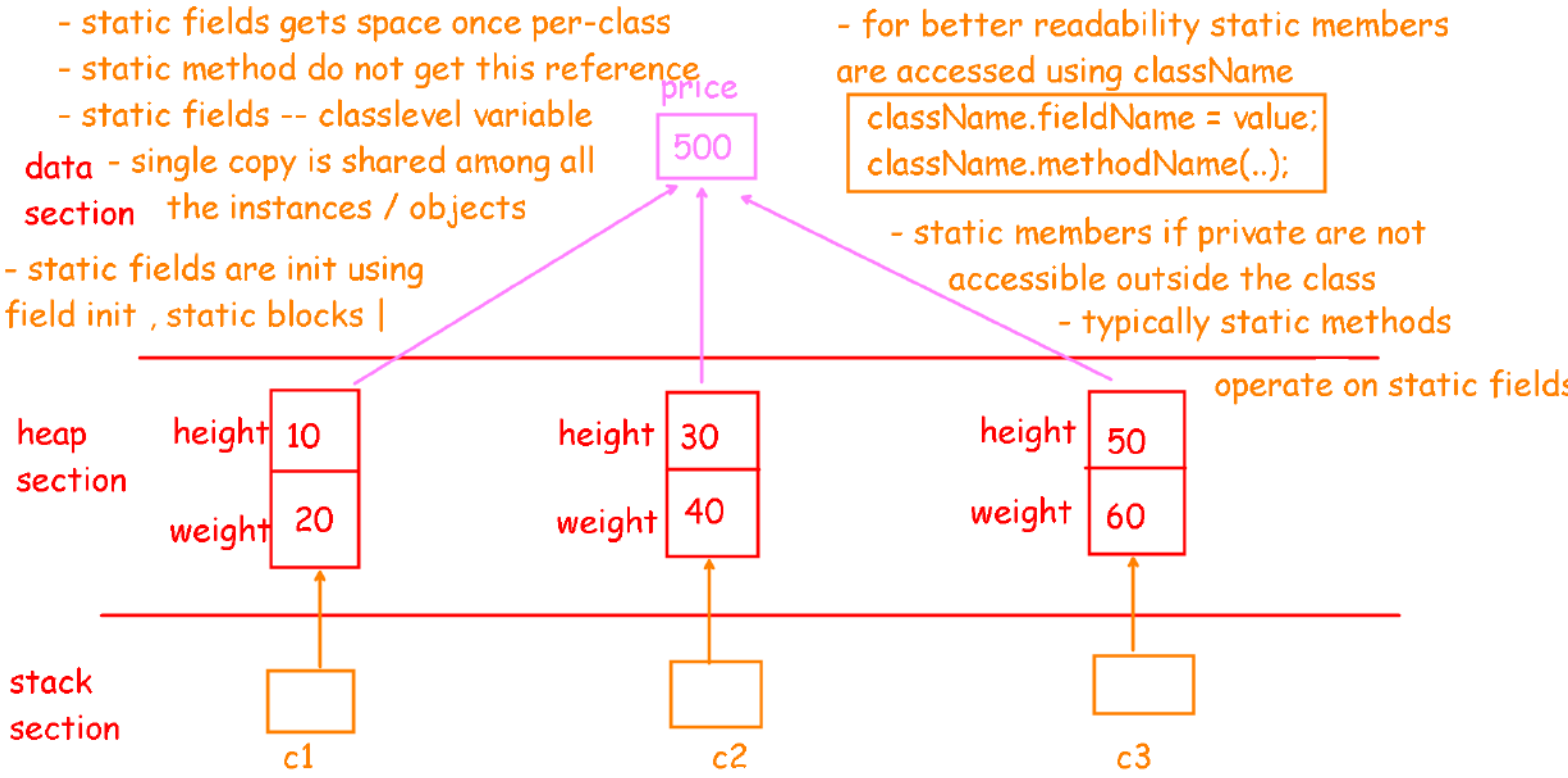


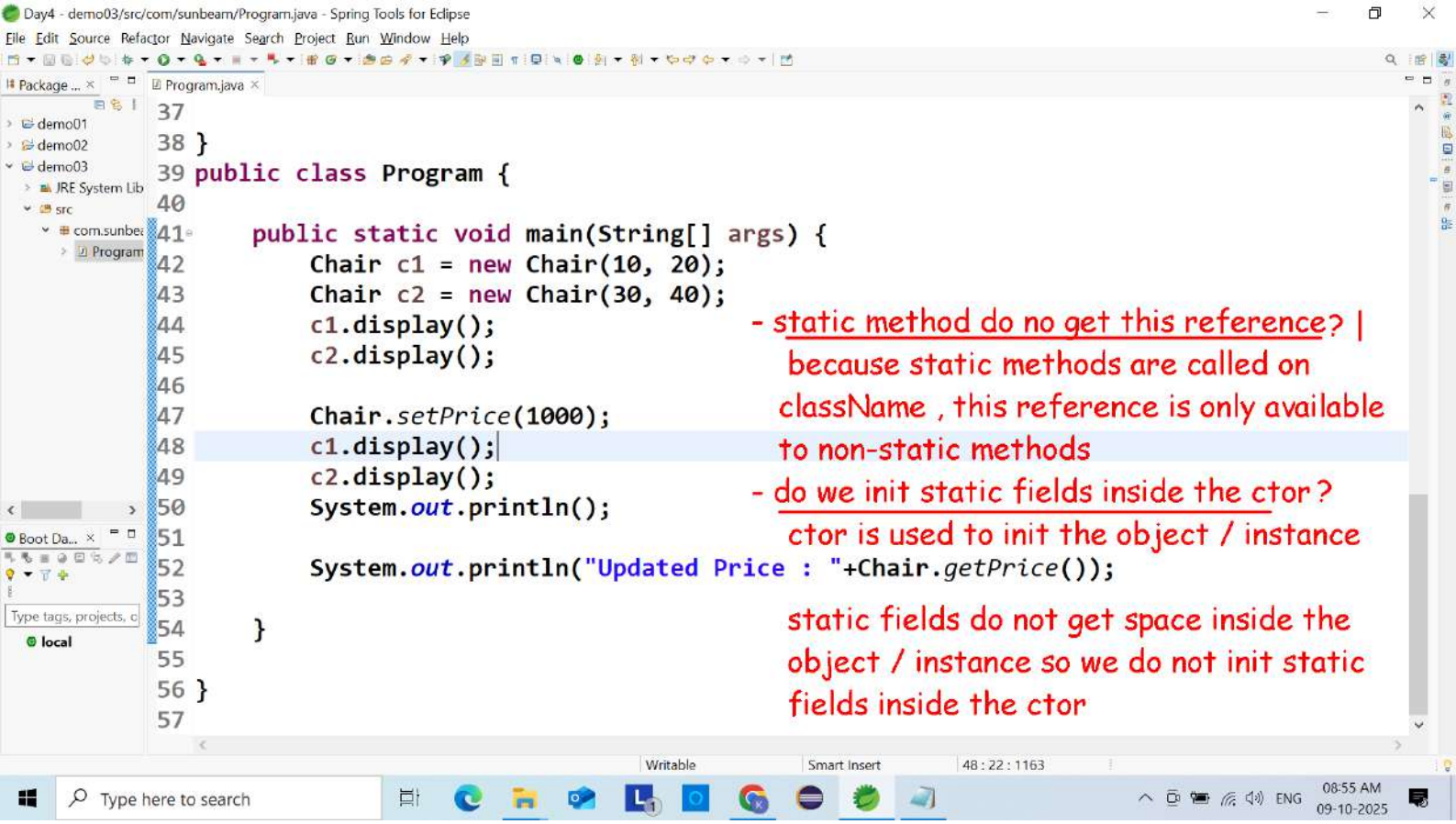
- Assigning subclass reference to super-class reference is called as upcasting

```
emp = (Employee) p; //downcasting
```

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


Day4 - demo02/src/com/sunbeam/Program.java - Spring Tools for Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
*Program.java
1 package com.sunbeam;
2 /* final field can be initialized using field initializer, obj initializer, or constructor.
3 * Once initialized, it cannot be modified again.
4 * However, final fields must be initialized (in any of the above options).
5 * If not, compiler raise error.
6 */
7 class Circle{
8
9 }
10 public class Program {
11
12 public static void main(String[] args) {
13 // TODO Auto-generated method stub
14
15 }
16
17 }
18
Final keyword
- field
- method
- class
- variable
final (init)
- field init
- obj init
- ctor
final fields must be initialiazed in any of the above options
if not compiler give error
Writable
Smart Insert
9 : 2 : 303
Type here to search
ENG
08:18 AM
09-10-2025

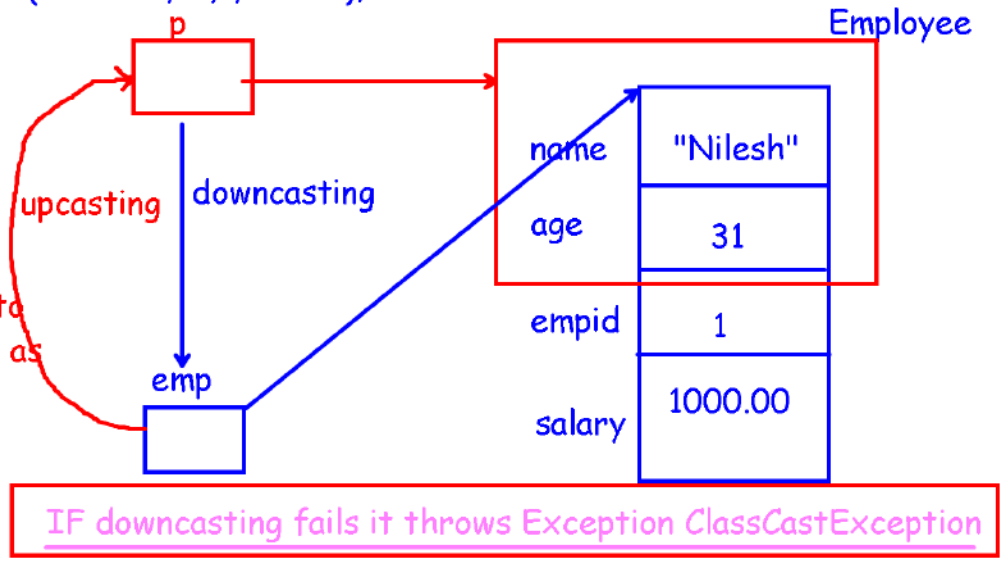

- static fields gets space once per-class
- static method do not get this reference
- static fields -- classlevel variable
data - single copy is shared among all
section the instances / objects
- static fields are init using
field init , static blocks |
price
500
- for better readability static members
are accessed using className
className.fieldName = value;
className.methodName(..);
- static members if private are not
accessible outside the class
- typically static methods
operate on static fields
heap
section
height
10
weight
20
height
30
weight
40
height
50
weight
60
stack
section
c1
c2
c3


Day4 - demo03/src/com/sunbeam/Program.java - Spring Tools for Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package ...
Program.java
demo01
demo02
demo03
JRE System Lib
src
com.sunbea
Program
Boot Da...
Type tags, projects, c
local
}
public class Program {
public static void main(String[] args) {
Chair c1 = new Chair(10, 20);
Chair c2 = new Chair(30, 40);
c1.display();
c2.display();
Chair.setPrice(1000);
c1.display();
c2.display();
System.out.println();
System.out.println("Updated Price : "+Chair.getPrice());
}
}
- static method do no get this reference? | because static methods are called on className , this reference is only available to non-static methods
- do we init static fields inside the ctor?
ctor is used to init the object / instance
static fields do not get space inside the object / instance so we do not init static fields inside the ctor
Writable
Smart Insert
48 : 22 : 1163
Type here to search
ENG
08:55 AM
09-10-2025

Person p = new Employee(..); // Upcasting

Employee emp = new Employee("Nilesh",31,1,1000.0);

sysout(emp.name); // OK
sysout(emp.age); // OK
sysout(emp.empid); // OK
sysout(emp.salary); // OK



Person p = (Person) emp;
//Assigning sub-class reference to super-class reference is called as upcasting

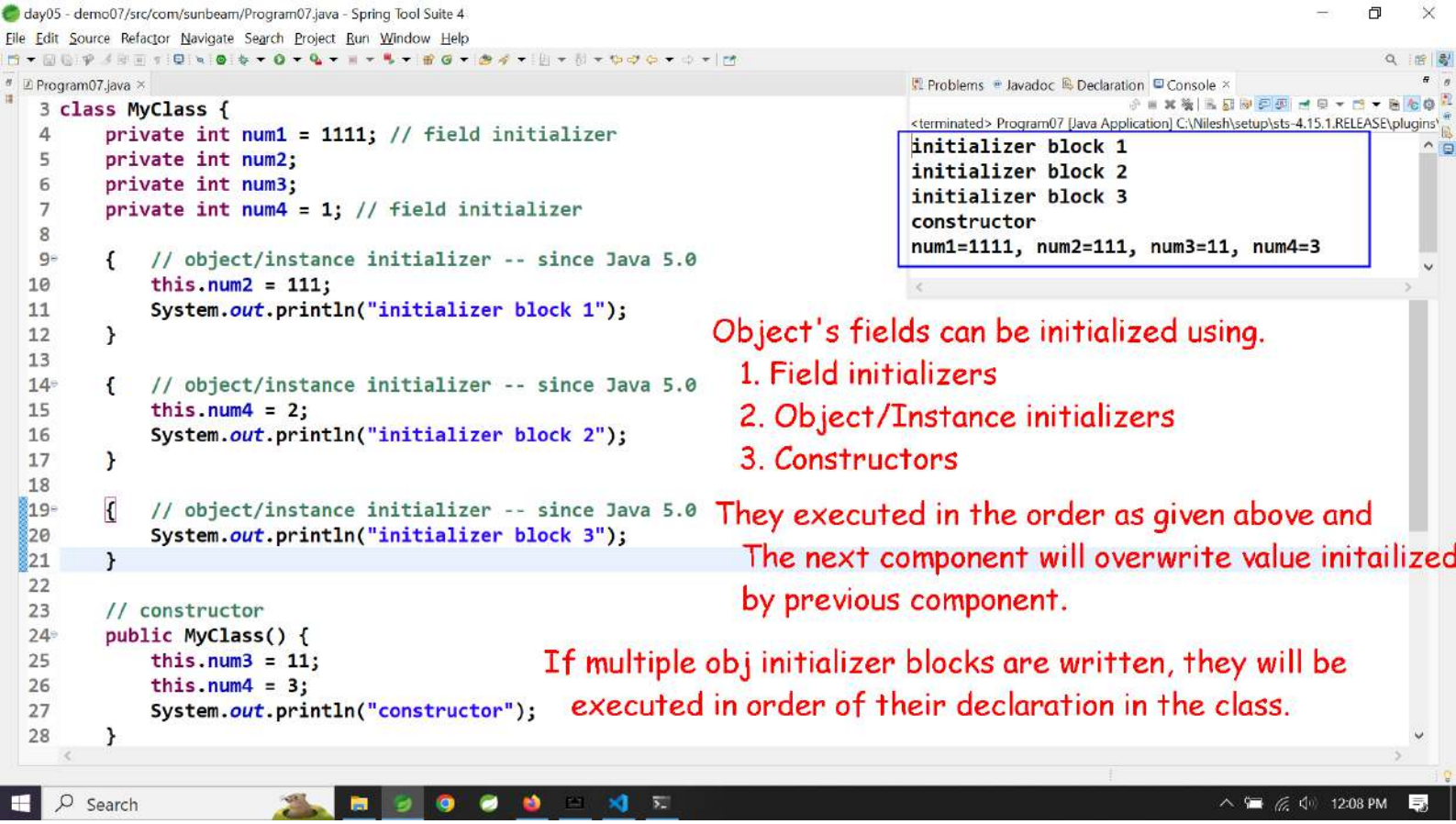
sysout(p.name); // OK
sysout(p.age); // OK
sysout(p.empid); // NOT OK
sysout(p.salary)); // NOT OK

IF downcasting fails it throws Exception ClassCastException

Person p = emp; // Upcasting

emp = (Employee) p; // downcasting
// Assigning super-class reference back to sub-class is called as dowcasting

sysout(emp.name); // OK
sysout(emp.age); // OK
sysout(emp.empid); // OK
sysout(emp.salary); // OK



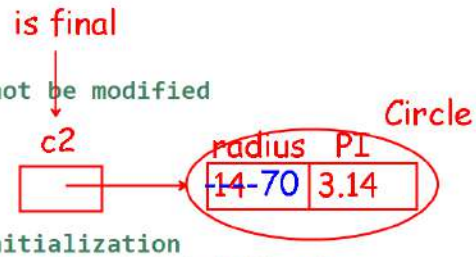

```
3 class MyClass {
4     private int num1 = 1111; // field initializer
5     private int num2;
6     private int num3;
7     private int num4 = 1; // field initializer
8
9     {   // object/instance initializer -- since Java 5.0
10        this.num2 = 111;
11        System.out.println("initializer block 1");
12    }
13
14    {   // object/instance initializer -- since Java 5.0
15        this.num4 = 2;
16        System.out.println("initializer block 2");
17    }
18
19    {   // object/instance initializer -- since Java 5.0
20        System.out.println("initializer block 3");
21    }
22
23    // constructor
24    public MyClass() {
25        this.num3 = 11;
26        this.num4 = 3;
27        System.out.println("constructor");
28    }
```

day05 - demo08/src/com/sunbeam/Program08.java - Spring Tool Suite 4

File Edit Source Refactor Navigate Search Project Run Window Help

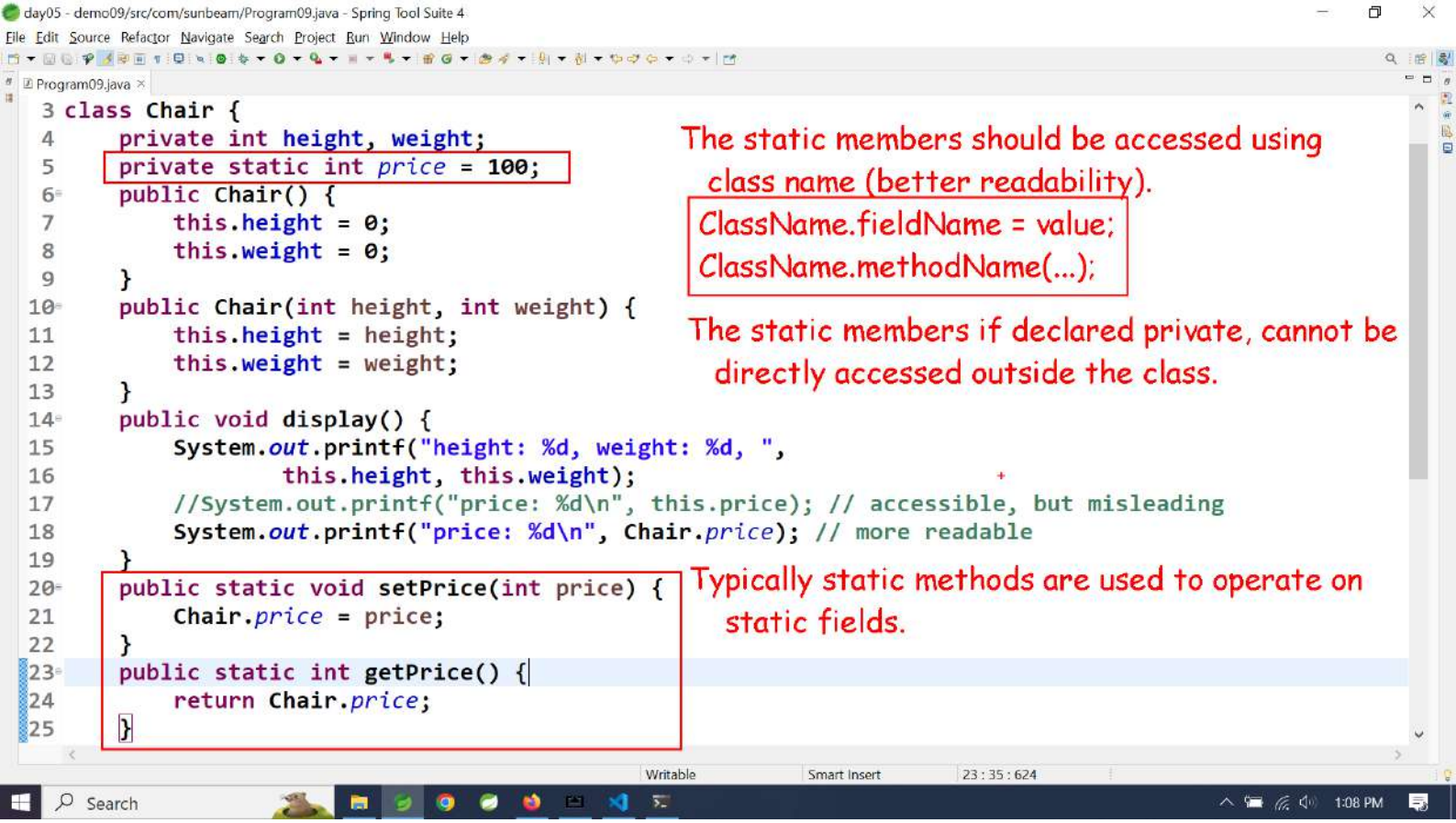
Program08.java

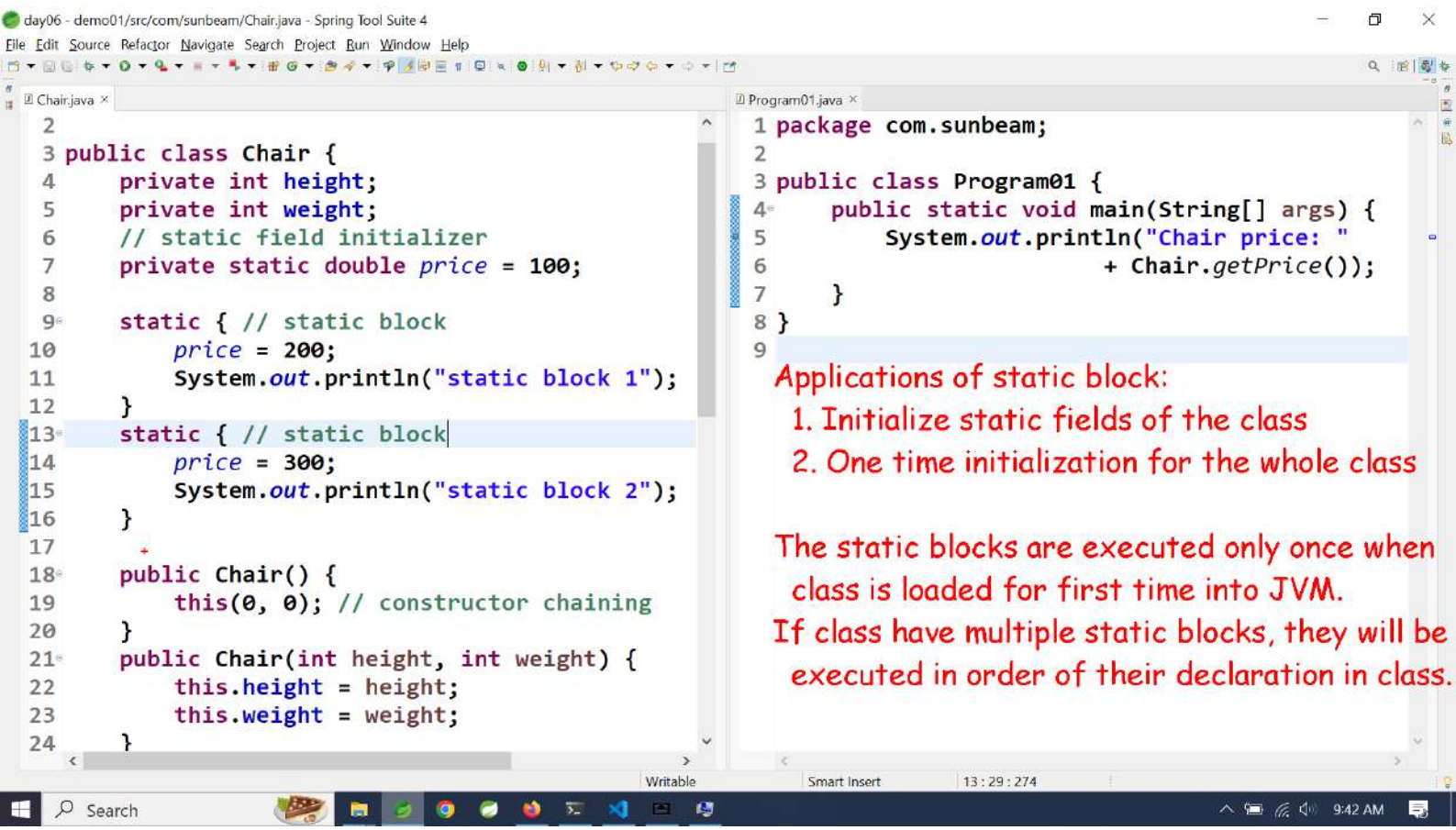
```
28          return 2 * PI * this.radius;
29      }
30  }
31
32  public class Program08 {
33      public static void main(String[] args) {
34          final double rad; // final var declaration
35          rad = 7; // final var initialization
36          //rad = 14; // compiler error: once initialized, cannot be modified
37          Circle c1 = new Circle(rad);
38          System.out.println("Area: " + c1.calcArea());
39          System.out.println("Peri: " + c1.calcPeri());
40
41          final double rad2 = 14; // final var declaration & initialization
42          final Circle c2 = new Circle(rad2); // final ref declaration & initialization
43          c2.setRadius(70); // obj state can be modified (as reference is final)
44          //c2 = null; // compiler error: reference once initialized, cannot be modified
45          System.out.println("Area: " + c2.calcArea());
46          System.out.println("Peri: " + c2.calcPeri());
47      }
48  }
49
50
```

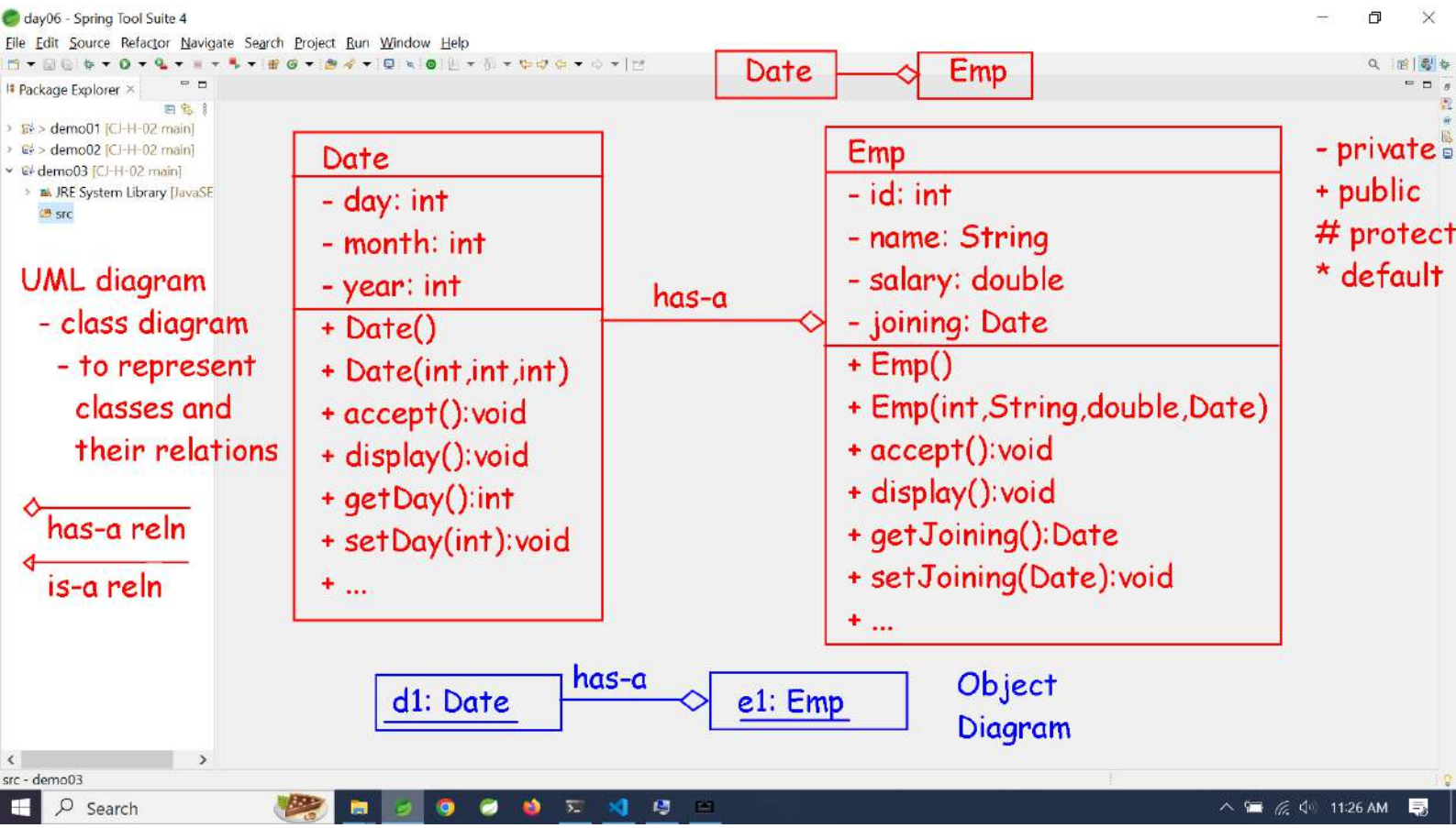


Writable Smart Insert 44 : 11 : 1490

Search 12:31 PM


day05 - demo09/src/com/sunbeam/Program09.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
Program09.java
3 class Chair {
4 private int height, weight;
5 private static int price = 100;
6 public Chair() {
7 this.height = 0;
8 this.weight = 0;
9 }
10 public Chair(int height, int weight) {
11 this.height = height;
12 this.weight = weight;
13 }
14 public void display() {
15 System.out.printf("height: %d, weight: %d, ",
16 this.height, this.weight);
17 //System.out.printf("price: %d\n", this.price); // accessible, but misleading
18 System.out.printf("price: %d\n", Chair.price); // more readable
19 }
20 public static void setPrice(int price) {
21 Chair.price = price;
22 }
23 public static int getPrice() {
24 return Chair.price;
25 }
The static members should be accessed using class name (better readability).
ClassName.fieldName = value;
ClassName.methodName(...);
The static members if declared private, cannot be directly accessed outside the class.
Typically static methods are used to operate on static fields.
Writable
Smart Insert
23 : 35 : 624
Search
1:08 PM


day06 - demo01/src/com/sunbeam/Chair.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
Chair.java
public class Chair {
private int height;
private int weight;
// static field initializer
private static double price = 100;
static { // static block
price = 200;
System.out.println("static block 1");
}
static { // static block
price = 300;
System.out.println("static block 2");
}
public Chair() {
this(0, 0); // constructor chaining
}
public Chair(int height, int weight) {
this.height = height;
this.weight = weight;
}
Program01.java
package com.sunbeam;
public class Program01 {
public static void main(String[] args) {
System.out.println("Chair price: "
+ Chair.getPrice());
}
}
Applications of static block:
1. Initialize static fields of the class
2. One time initialization for the whole class
The static blocks are executed only once when class is loaded for first time into JVM.
If class have multiple static blocks, they will be executed in order of their declaration in class.
Writable
Smart Insert
13 : 29 : 274
Search
9:42 AM

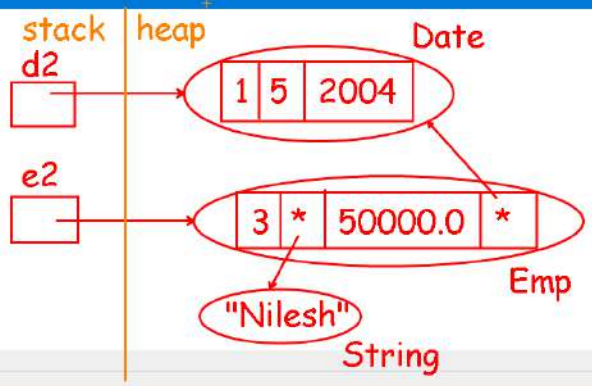

day06 - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
demo01 [CJ-H-02 main]
demo02 [CJ-H-02 main]
demo03 [CJ-H-02 main]
JRE System Library [JavaSE
src
Date
Emp
UML diagram
- class diagram
- to represent classes and their relations
has-a reln
is-a reln
Date
- day: int
- month: int
- year: int
+ Date()
+ Date(int,int,int)
+ accept():void
+ display():void
+ getDay():int
+ setDay(int):void
+ ...
has-a
Emp
- id: int
- name: String
- salary: double
- joining: Date
+ Emp()
+ Emp(int,String,double,Date)
+ accept():void
+ display():void
+ getJoining():Date
+ setJoining(Date):void
+ ...
- private
+ public
# protect
* default
d1: Date
has-a
e1: Emp
Object Diagram
src - demo03
Search
11:26 AM

day06 - demo03/src/com/sunbeam/Program03.java - Spring Tool Suite 4

File Edit Source Refactor Navigate Search Project Run Window Help

Date.java Emp.java Program03.java

```
1  package com.sunbeam;
2
3  public class Program03 {
4      public static void main(String[] args) {
5          Emp e1 = new Emp();
6          e1.display();
7          System.out.println();
8
9          Date d2 = new Date(1, 5, 2004);
10         Emp e2 = new Emp(3, "Nilesh Ghule", 50000.0, d2);
11         e2.display();
12         System.out.println();
13
14         Emp e3 = new Emp();
15         e3.accept();
16         e3.display();
17     }
18 }
19
```

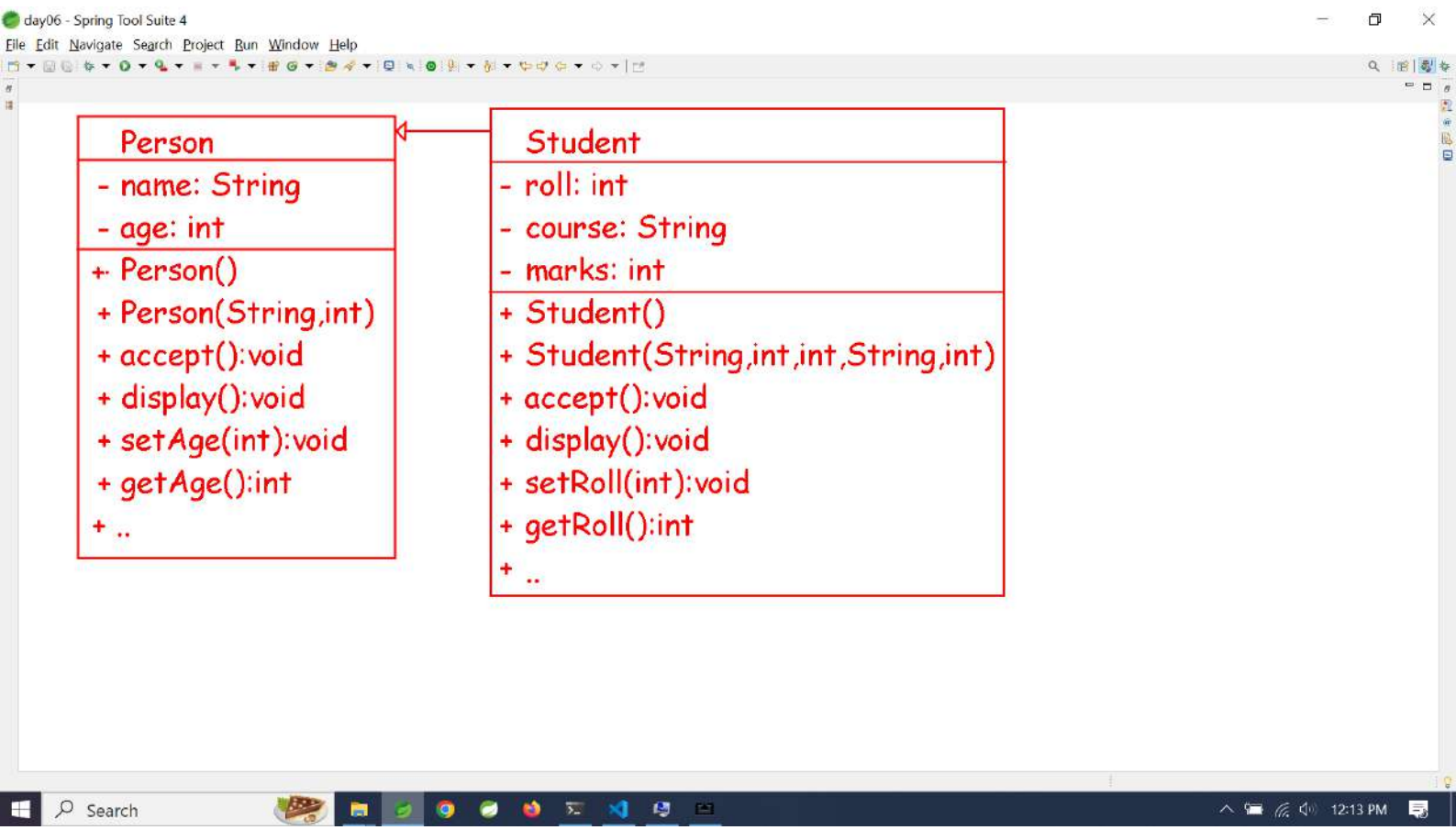


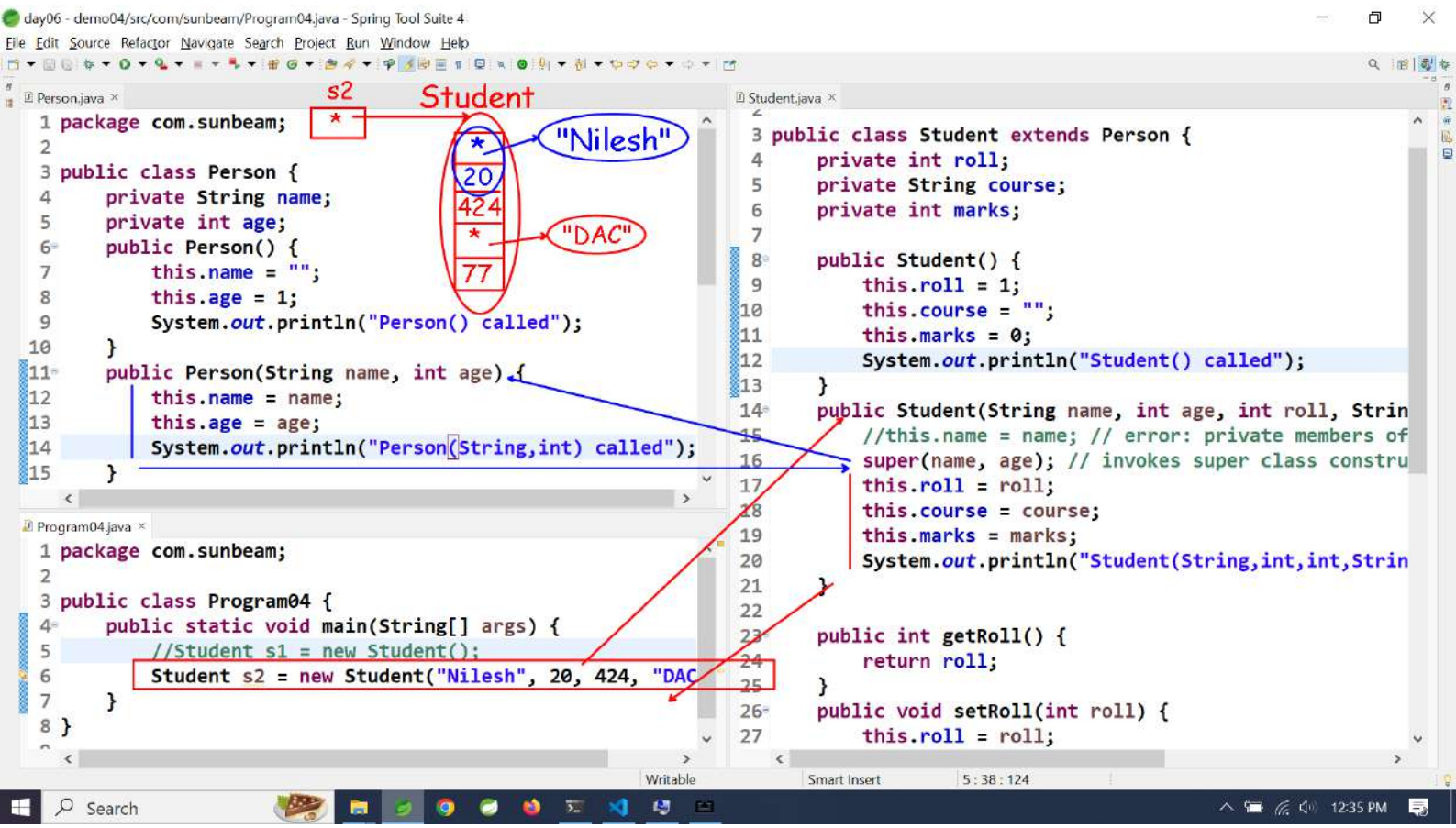
Problems Javadoc Declaration Console

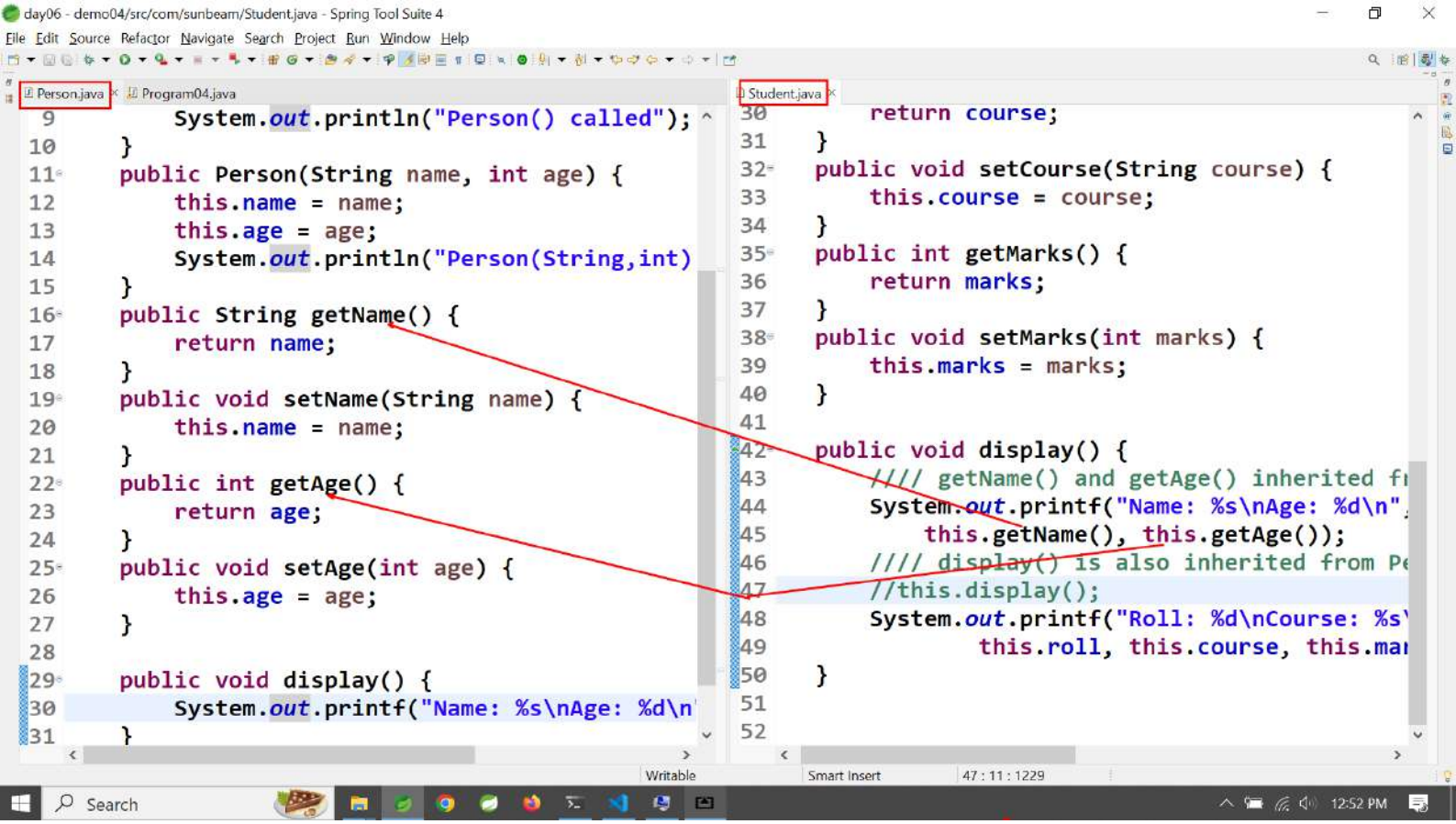
<terminated> Program03 [Java Application] C:\Nilesh\setup\sts

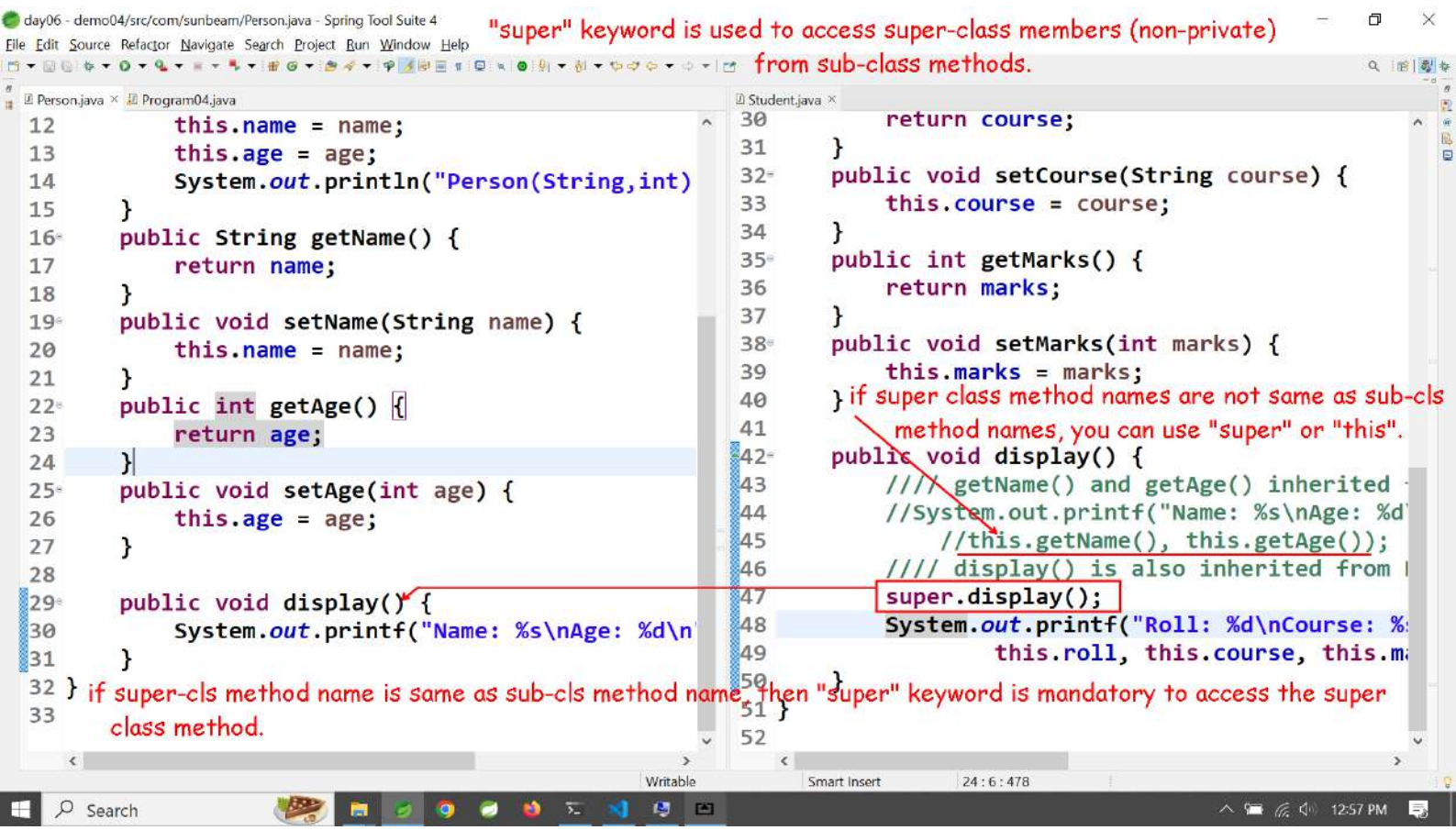
```
Id: 0
Name:
Sal: 0.000000
Joining Date: 1-1-2000

Id: 3
Name: Nilesh Ghule
Sal: 50000.000000
Joining Date: 1-5-2004

Id: 7
Name: James Bond
Sal: 80000
Joining Date (day month yea
Id: 7
Name: James Bond
Sal: 80000.000000
Joining Date: 1-1-1900
```

Search 12:00 PM


day06 - Spring Tool Suite 4
File Edit Navigate Search Project Run Window Help
Person
- name: String
- age: int
+ Person()
+ Person(String,int)
+ accept():void
+ display():void
+ setAge(int):void
+ getAge():int
+ ..
Student
- roll: int
- course: String
- marks: int
+ Student()
+ Student(String,int,int,String,int)
+ accept():void
+ display():void
+ setRoll(int):void
+ getRoll():int
+ ..
Search
12:13 PM

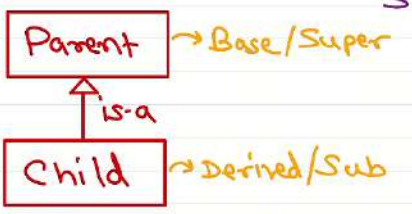

day06 - demo04/src/com/sunbeam/Program04.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
Person.java
Student.java
Program04.java
s2
Student
"Nilesh"
"DAC"
20
424
77
package com.sunbeam;
public class Person {
private String name;
private int age;
public Person() {
this.name = "";
this.age = 1;
System.out.println("Person() called");
}
public Person(String name, int age) {
this.name = name;
this.age = age;
System.out.println("Person(String,int) called");
}
public class Student extends Person {
private int roll;
private String course;
private int marks;
public Student() {
this.roll = 1;
this.course = "";
this.marks = 0;
System.out.println("Student() called");
}
public Student(String name, int age, int roll, Strin
//this.name = name; // error: private members of
super(name, age); // invokes super class constru
this.roll = roll;
this.course = course;
this.marks = marks;
System.out.println("Student(String,int,int,Strin
}
public int getRoll() {
return roll;
}
public void setRoll(int roll) {
this.roll = roll;
public class Program04 {
public static void main(String[] args) {
//Student s1 = new Student();
Student s2 = new Student("Nilesh", 20, 424, "DAC
}
}
Writable
Smart Insert
5 : 38 : 124
Search
12:35 PM


day06 - demo04/src/com/sunbeam/Student.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
Person.java
Program04.java
Student.java
System.out.println("Person() called");
}
public Person(String name, int age) {
this.name = name;
this.age = age;
System.out.println("Person(String,int)
}
public String getName() {
return name;
}
public void setName(String name) {
this.name = name;
}
public int getAge() {
return age;
}
public void setAge(int age) {
this.age = age;
}
public void display() {
System.out.printf("Name: %s\nAge: %d\n
}
return course;
}
public void setCourse(String course) {
this.course = course;
}
public int getMarks() {
return marks;
}
public void setMarks(int marks) {
this.marks = marks;
}
public void display() {
//// getName() and getAge() inherited fr
System.out.printf("Name: %s\nAge: %d\n",
this.getName(), this.getAge());
//// display() is also inherited from Pe
//this.display();
System.out.printf("Roll: %d\nCourse: %s
this.roll, this.course, this.mar
}
Writable
Smart Insert
47 : 11 : 1229
Search
12:52 PM


day06 - demo04/src/com/sunbeam/Person.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
"super" keyword is used to access super-class members (non-private) from sub-class methods.
Person.java
Program04.java
Student.java
this.name = name;
this.age = age;
System.out.println("Person(String,int)
}
public String getName() {
return name;
}
public void setName(String name) {
this.name = name;
}
public int getAge() {
return age;
}
public void setAge(int age) {
this.age = age;
}
public void display() {
System.out.printf("Name: %s\nAge: %d\n
}
} if super-cls method name is same as sub-cls method name, then "super" keyword is mandatory to access the super class method.
return course;
}
public void setCourse(String course) {
this.course = course;
}
public int getMarks() {
return marks;
}
public void setMarks(int marks) {
this.marks = marks;
}
if super class method names are not same as sub-cls method names, you can use "super" or "this".
public void display() {
//// getName() and getAge() inherited
//System.out.printf("Name: %s\nAge: %d
//this.getName(), this.getAge());
//// display() is also inherited from
super.display();
System.out.printf("Roll: %d\nCourse: %
this.roll, this.course, this.m
}
}
Writable
Smart Insert
24 : 6 : 478
Search
12:57 PM

Inheritance

To avoid ambiguity errors due to multiple inheritance, Java lanaguage doesn't have support for multiple implementation(class) inheritance. However, Java allows multiple interface inheriance.



All members of parent class are inherited to the child class.

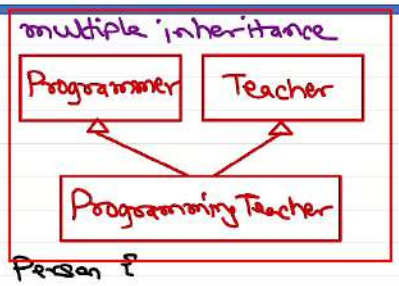
Parent/Super class : generalized

inheritance

Child/Sub class : Specialized

super keyword:

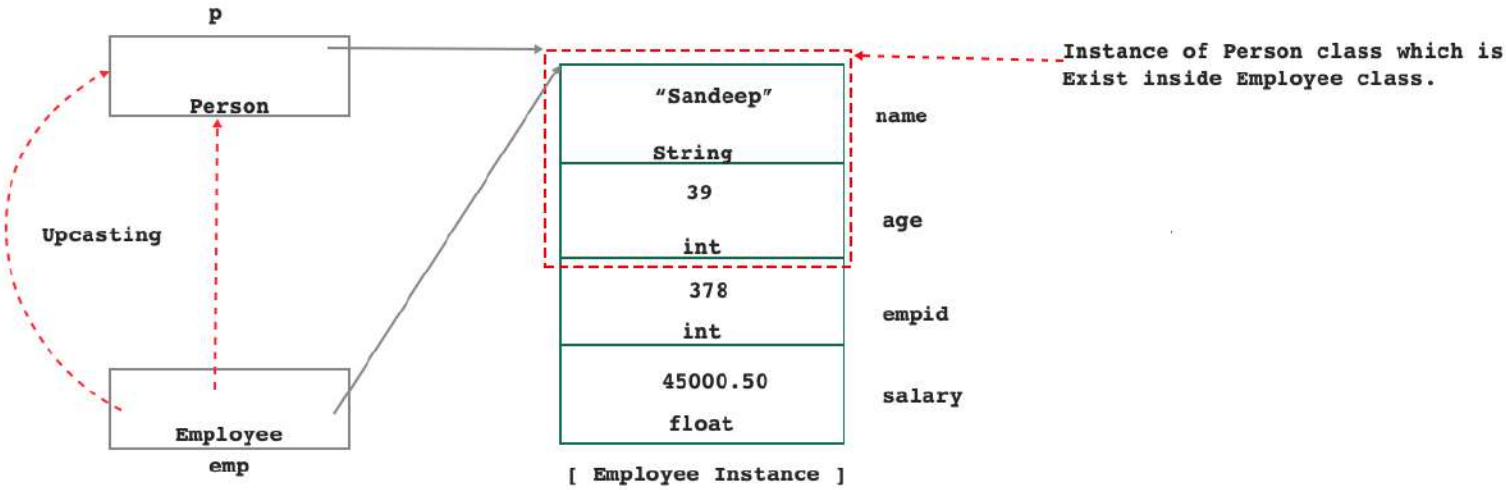
1. invoke super class ctor from sub class ctor.
2. access super class members from sub class non-static methods.

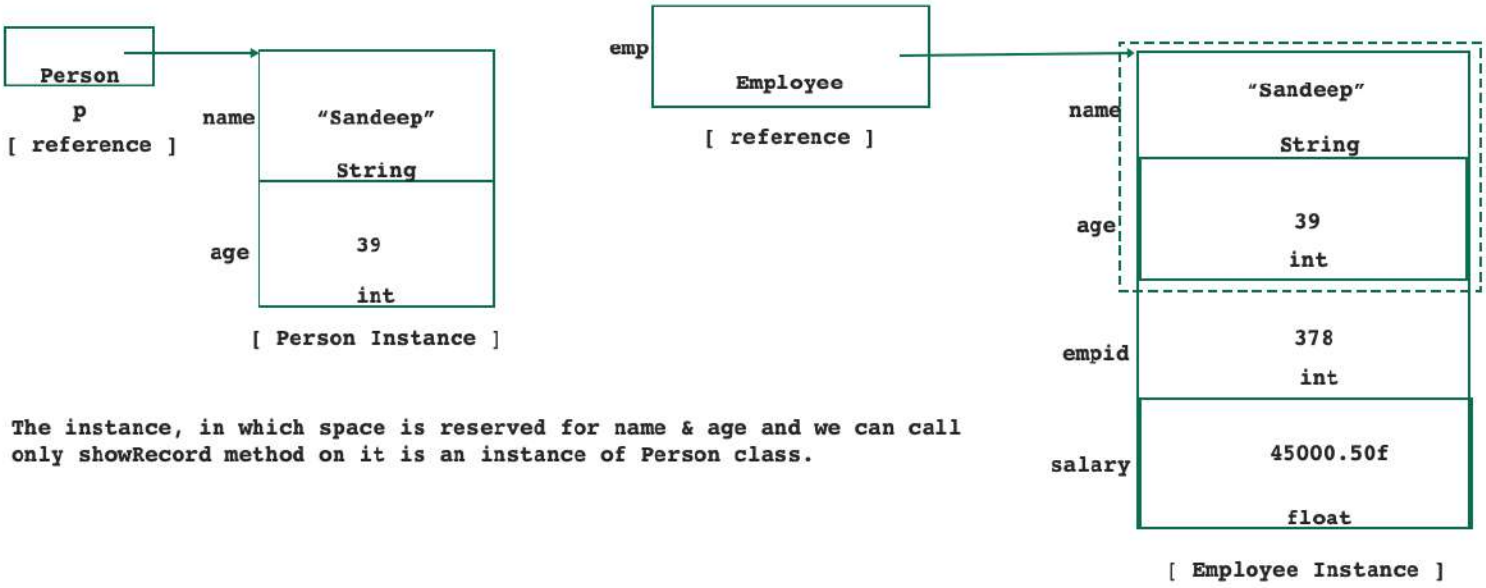



```
class Person {
    =
    public void display() {
    =
    }
};
class Student extends Person {
    =
    public void display() {
    super.display();
    =
    }
};
main():
    Student s1= new Student();
    s1.display();
    // cannot call Person.display on Student ref → Method shadowing
```

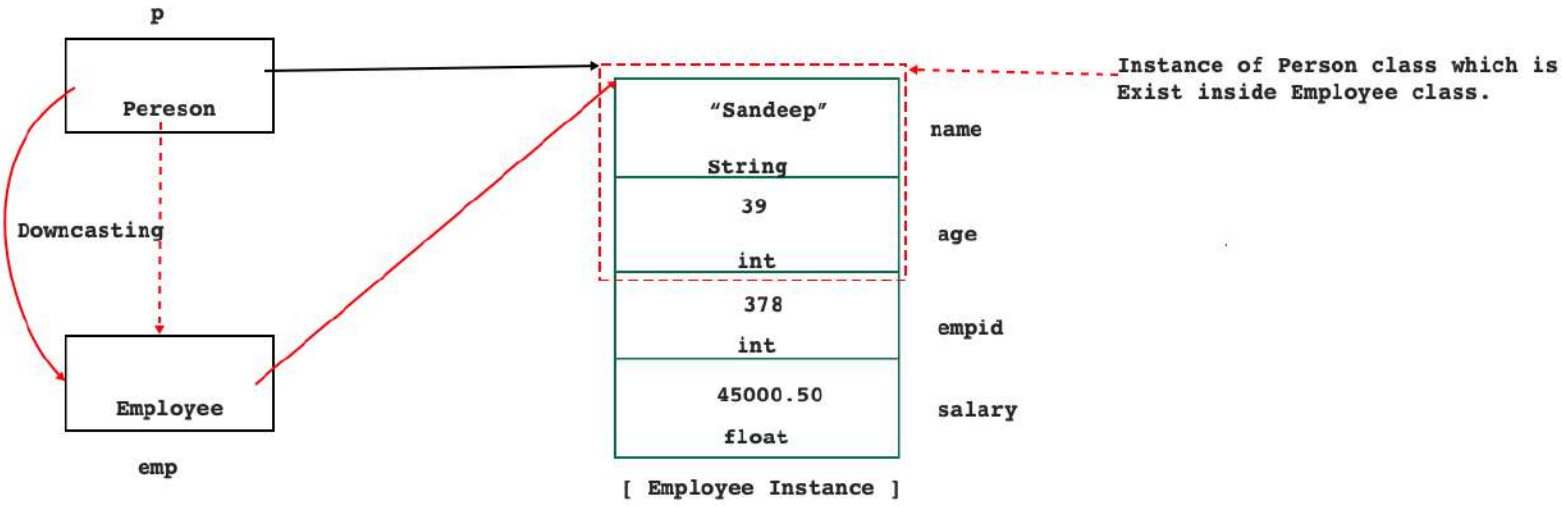
In C++:

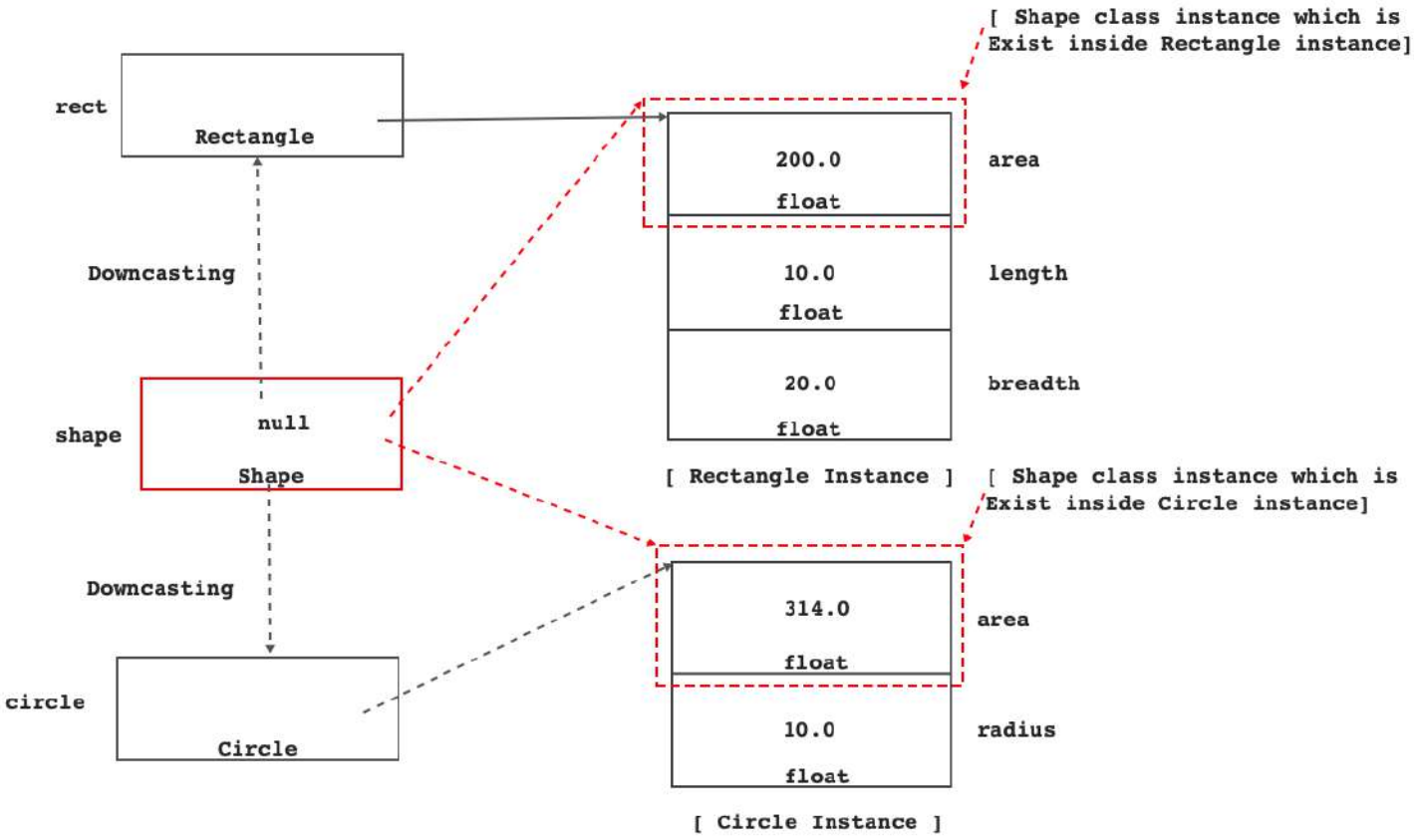
```
class Programmer {
  public:
    void display() { ... }
};
class Teacher {
  public:
    void display() { ... }
};
class ProgrammingTeacher:
    public Programmer, Teacher {
  // ...
};
main():
ProgrammingTeacher obj;
obj.display(); // ambiguity error
```


p
Person
Upcasting
Employee
emp
"Sandeep"
String
name
39
int
age
378
int
empid
45000.50
float
salary
[Employee Instance]
Instance of Person class which is
Exist inside Employee class.



The instance, in which space is reserved for name & age and we can call only showRecord method on it is an instance of Person class.


p
Pereson
Downcasting
Employee
emp
"Sandeep"
String
name
39
int
age
378
int
empid
45000.50
float
salary
[Employee Instance]
Instance of Person class which is
Exist inside Employee class.


[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]