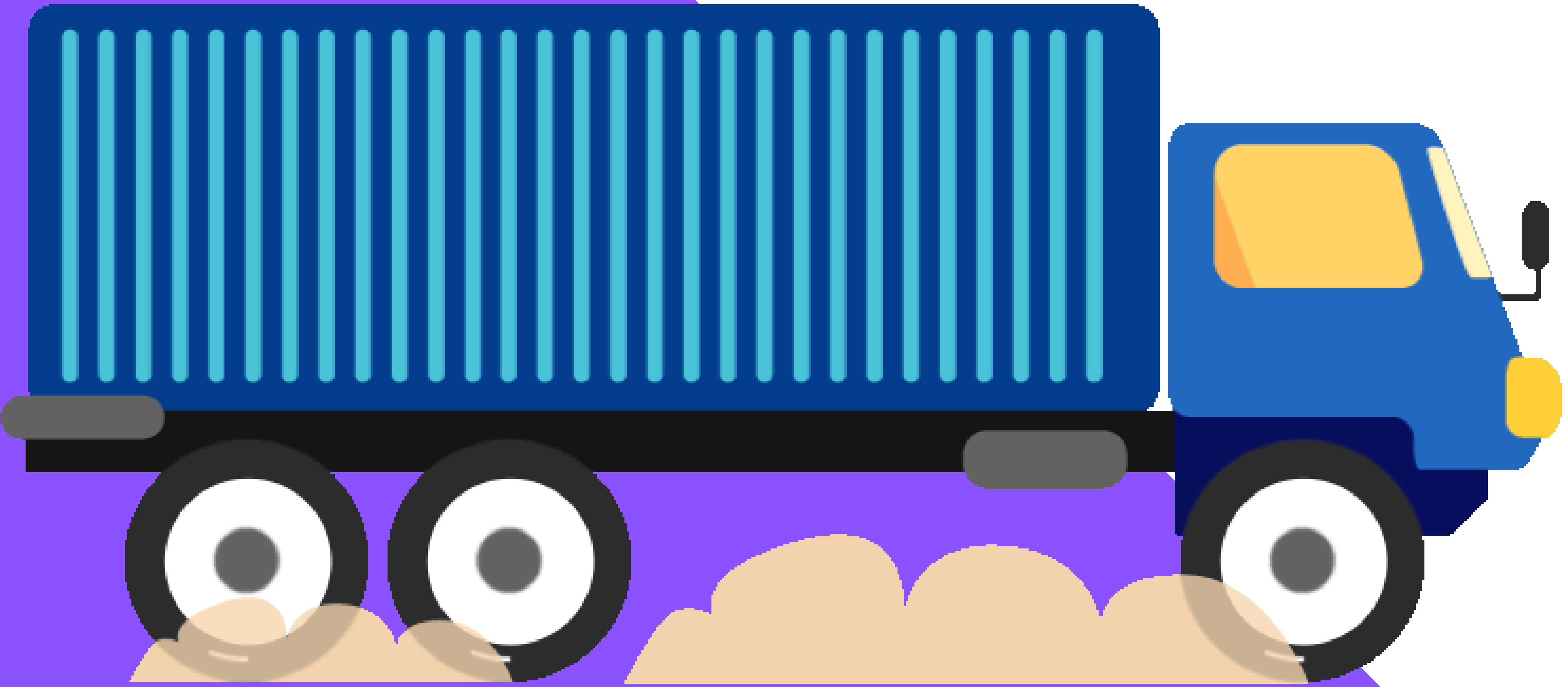
OVERLODING



overloading

Two methods are said to be overload if and only if both having the same name but different argument types.

Overloading

At compile-time, java knows which method to call by checking the method signatures. (method name + argument)

So this is called compile-time polymorphism or static or early binding.

simple program no overloading

```
public class Test {

        public static void main(String[] args) {
                sum(2,5);
         }

        private static void sum(int i, int j) {
                System.out.println(i+j);
        }
}
```

```
public class Test{

    public static void main(String[] args) {
            sum(2 , 5);
            sum(2.5 , 5.5);
    }

    private static void sum(double d, double e) {
            System.out.println(d+e);
    }

    private static void sum(int i, int j) {
            System.out.println(i+j);
    }

}
```

While resolving overloaded methods exact match will always get high priority,

output

```
7
8.0
```

case 1 - Promotion case

@shivam

```
public class Test{

        public static void main(String[] args) {
                sum(2 , 5);

        }

        private static void sum(double d, double e) {
                System.out.println(d+e);
        }
}
```

no exact match but

no CT error why ??

output
```
7.0
```

because

In overloading if compiler is unable to find the method with exact match

we compiler promotes the argument to the next level

and checks whether the matched method is available

if matched, ok ...

if the matched method is not available

then we will get compile time error.

overloading

on the basis of this chart promotion is done

$$\begin{tikzcd}
byte \arrow[r] & short \arrow[dr] & & & & \\
 & & int \arrow[r] & long \arrow[r] & float \arrow[r] & double \\
 & char \arrow[ur] & & & &
\end{tikzcd}$$

case 1

so int argument
promoted to
double argument
2 -> 2.0
5 -> 5.0

```
public class Test{

        public static void main(String[] args) {
                sum(2 , 5);

        }

          private static void sum(double d, double e) {
                System.out.println(d+e);
          }
}
```

output
7.0

ERROR

```
public class Test{

        public static void main(String[] args) {

                sum(2.5 , 5.5);
        }

        private static void sum(int i, int j) {
                System.out.println(i+j);
        }

}
```

promotion of argument is possible not demotion !

output - error

case 2 - Playing with sequence

@shivam

```
public class Test{

    public static void main(String[] args) {
            sum(10.5 f , 10);
            sum(10 , 5.5 f);
    }

    private static void sum(int i, float j) {
            System.out.println(i+j);
    }

    private static void sum(float e , int f) {
            System.out.println(e+f);
    }
}
```

Exact match will get higher priority.

if exact match not found promot argument

output

```
20.5
15.5
```

ERROR

@shivam

```
public class Test{

public static void main(String[] args) {
            sum(10 , 10);


 }

  private static void sum(float f , int i) {
            System.out.println(f+i);
  }

  private static void sum(int i, float f) {
            System.out.println(i+f);
  }
```

Since exact match not found

if java

promot lst aargumet

then this will match

if java

promot 2nd agrumnet

then this will match

we will get

CE:reference to methodOne

is ambiguous

and its obvious

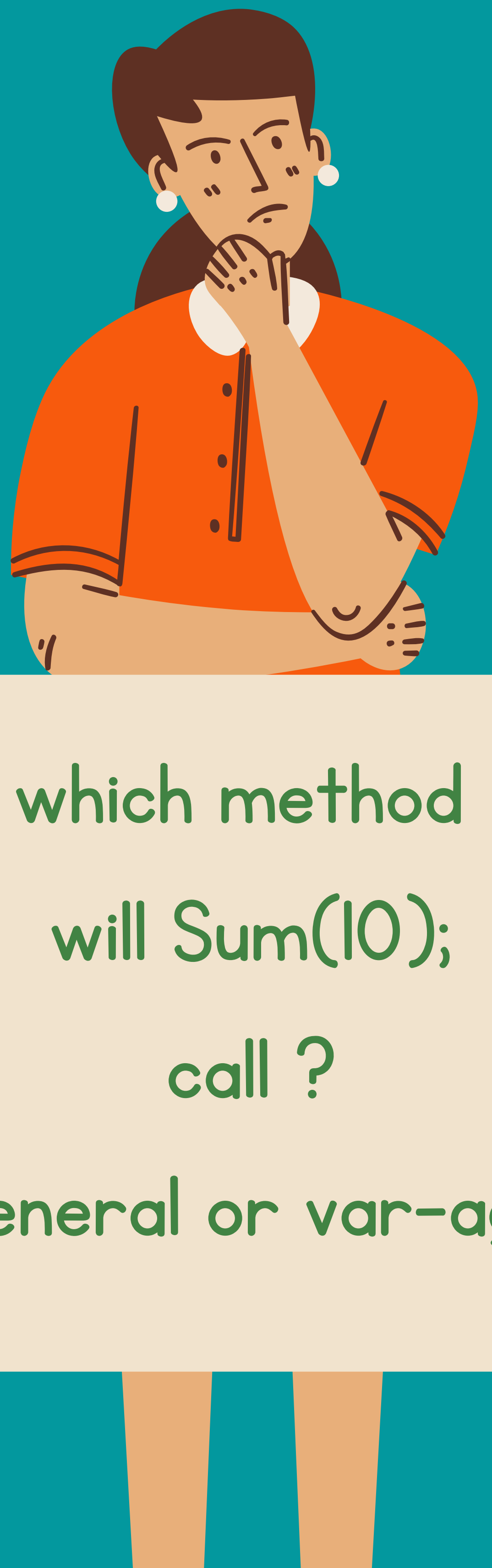
```
public class Test{

public static void main(String[] args) {
            sum(10 , 10);


 }

  private static void sum(float f , int i) {
            System.out.println(f+i);
  }

  private static void sum(int i, float f) {
            System.out.println(i+f);
  }
```

output - error

case 3 intro to var-arg

@shivam

```
public class Test{

    public static void main(String[] args) {
            sum(10);
            sum();
            sum(10,20);
    }

    private static void sum(int i) {
            System.out.println("general");
    }

    private static void sum(int ... i ) {
              System.out.println("var-arg");
    }
}
```



which method

will Sum(10);

call ?

general or var-agr

case 3 intro to var-arg

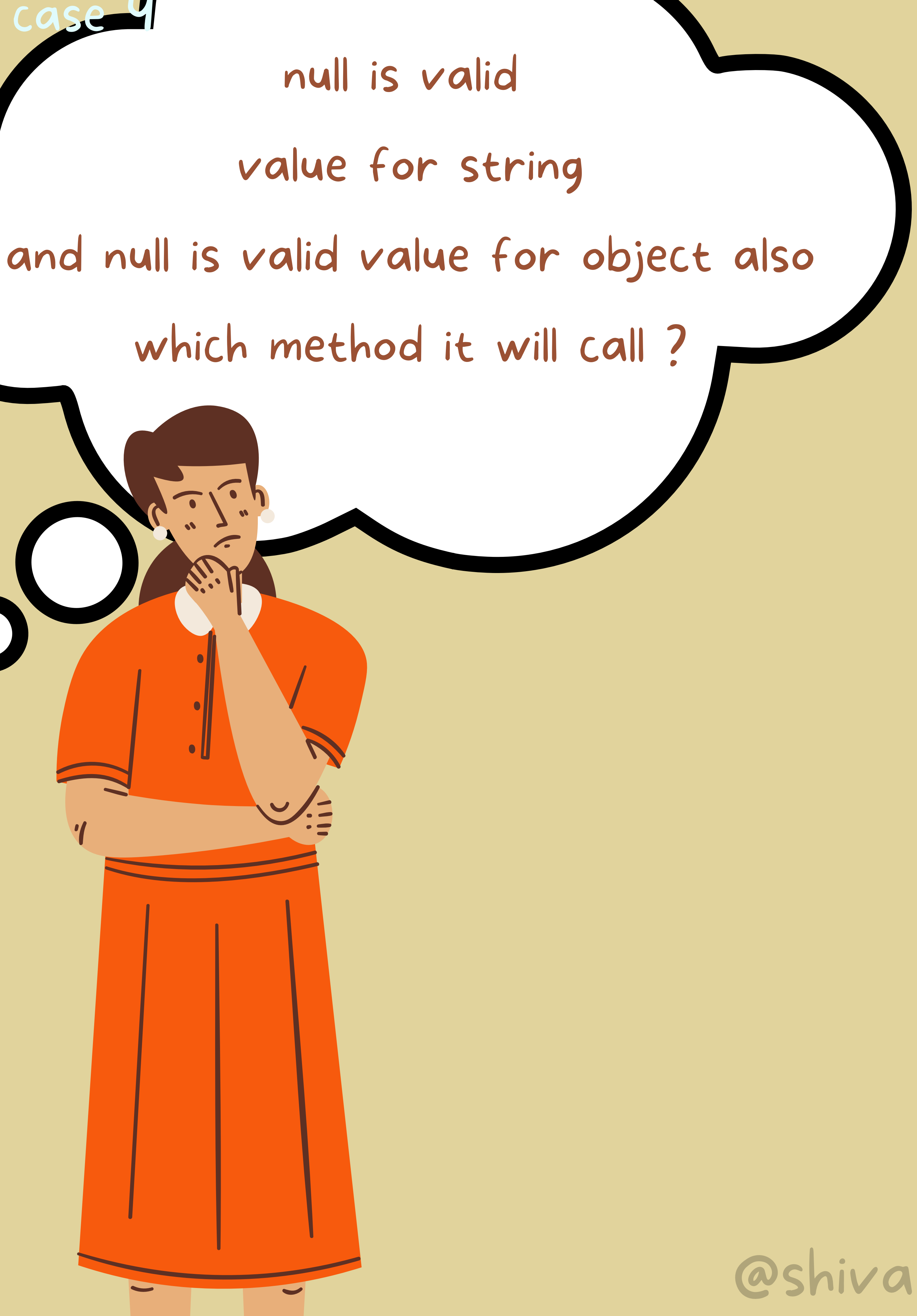
@shivam

```
public class Test{

    public static void main(String[] args) {
            sum(10);
            sum();
            sum(10,20);
    }

    private static void sum(int i) {
            System.out.println("general");
    }

    private static void sum(int ... i ) {
            System.out.println("var-arg");
    }
}
```

if no other method matched then only var-arg method will get chance for execution upto here our argument used to be actual value

@shivam

case 4


null is valid
value for string
and null is valid value for object also
which method it will call ?

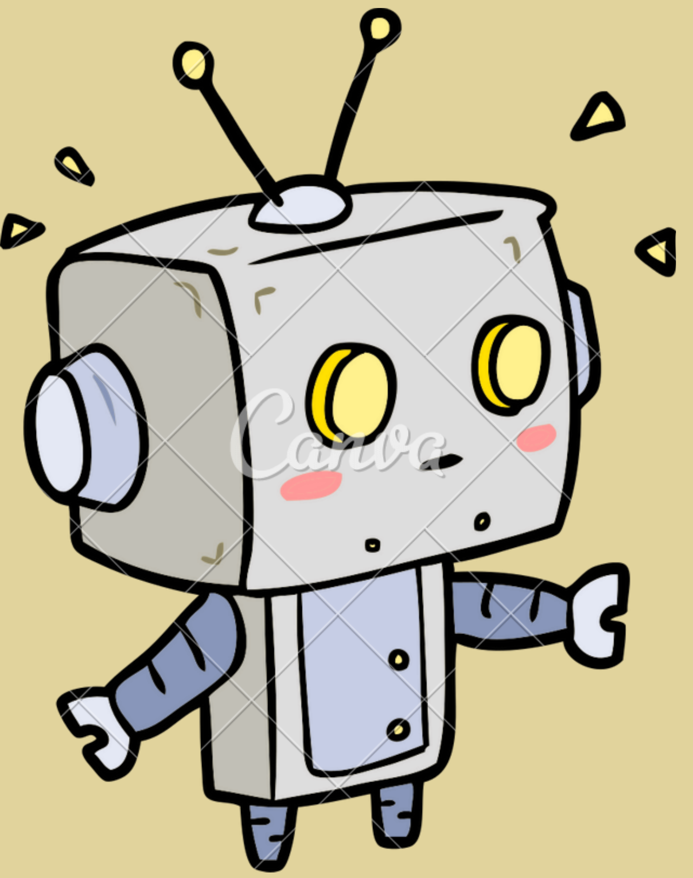
```
public class Sunny {

    public static void main(String[] args) {
            call (null);
    }

    private static void call(Object o) {
            System.out.println("object " + o);
    }

    private static void call(String s) {
            System.out.println("string " + s);
    }

}
```

@shivam

what is relation between object and string ?

$$\begin{tikzcd}\text{Object} \\ \text{String} \arrow[u]\end{tikzcd}$$

String class is child class of object



java said - if child can do that work

why to call parent ? ....lets call child class

child be like

ya, even my papa can fill water
bottle and place it in fridge
but i am child and
i am supposed to fill water bottle

Child

@shivam

```
public class Sunny {

    public static void main(String[] args) {
            call (null);
    }

    private static void call(Object o) {
            System.out.println("object " );
    }

    private static void call(String s) {
            System.out.println("string " );
    }

}
```

String method will

get called

output - string

case 5

Exact match is not there

we have parent class

and one sibling class

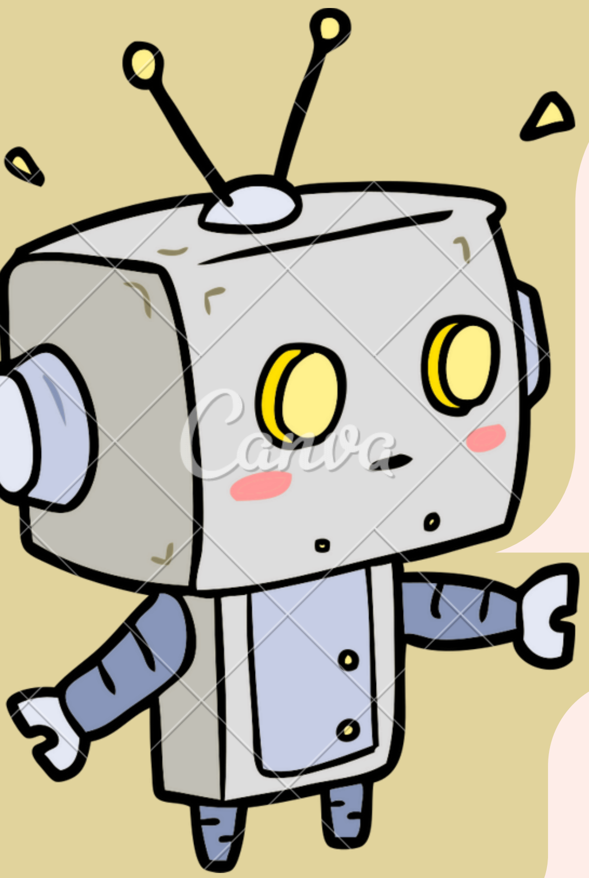
in primitive we were

promoting value...

which method will it call ?

```
public class Sunny {

    public static void main(String[] args) {
            Integer i = new Integer(10);
          method(i);
      }

    public static void method(Object d) {
              System.out.println("object");
     }

    public static void method(Double d) {
              System.out.println("double");
     }

}
```

$$\begin{tikzcd}
\text{Object} \\
\text{Number} \arrow[u] \\
\text{Integer} \arrow[u]
\end{tikzcd}$$



java said- if exact match is not found

then call its parent class !

Integer's parent is Number that is also not present

java said - call parent of Number class

```
public class Sunny {

    public static void main(String[] args) {
            Integer i = new Integer(10);
            method(i);
    }
    public static void method(Object d) {
            System.out.println("object");
    }

    public static void method(Double d) {
            System.out.println("double");
    }
}
```

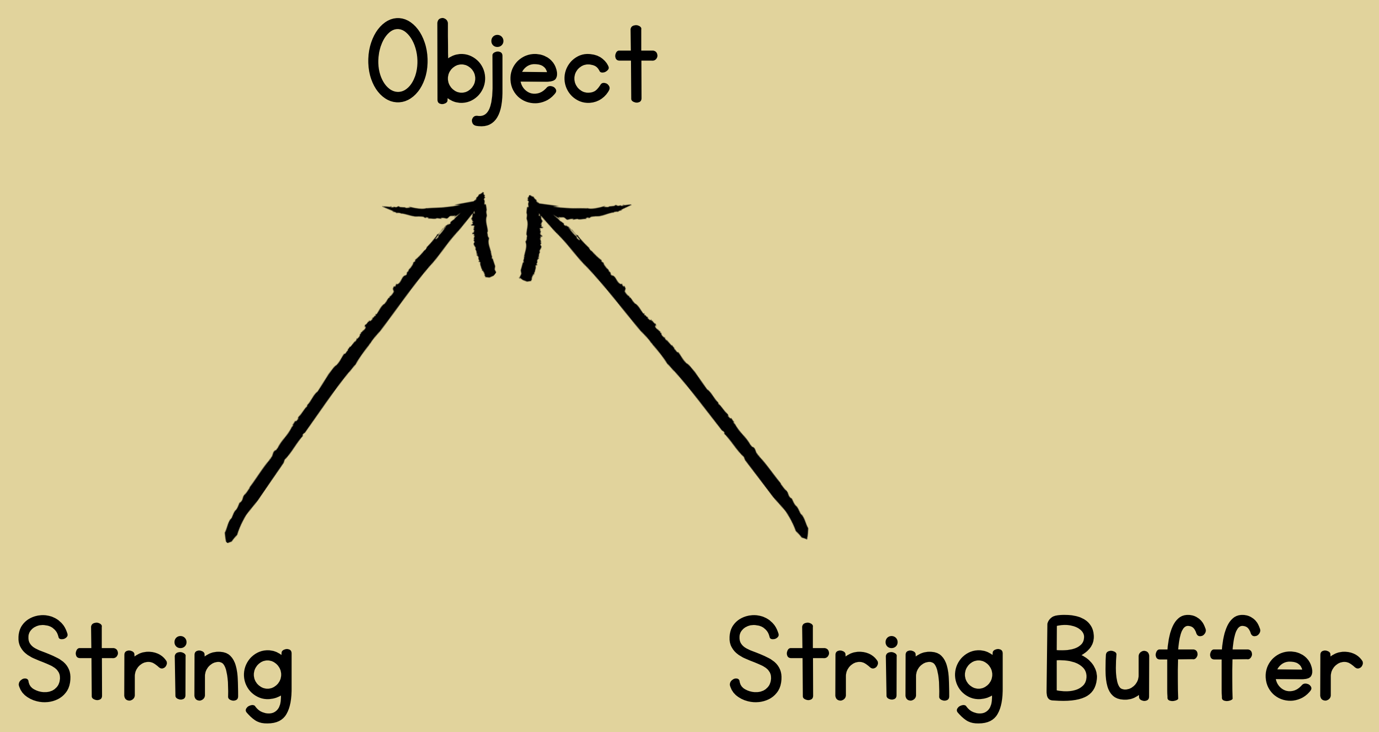
Object method will be called

promotion of Integer will not happen

output - object

case 6

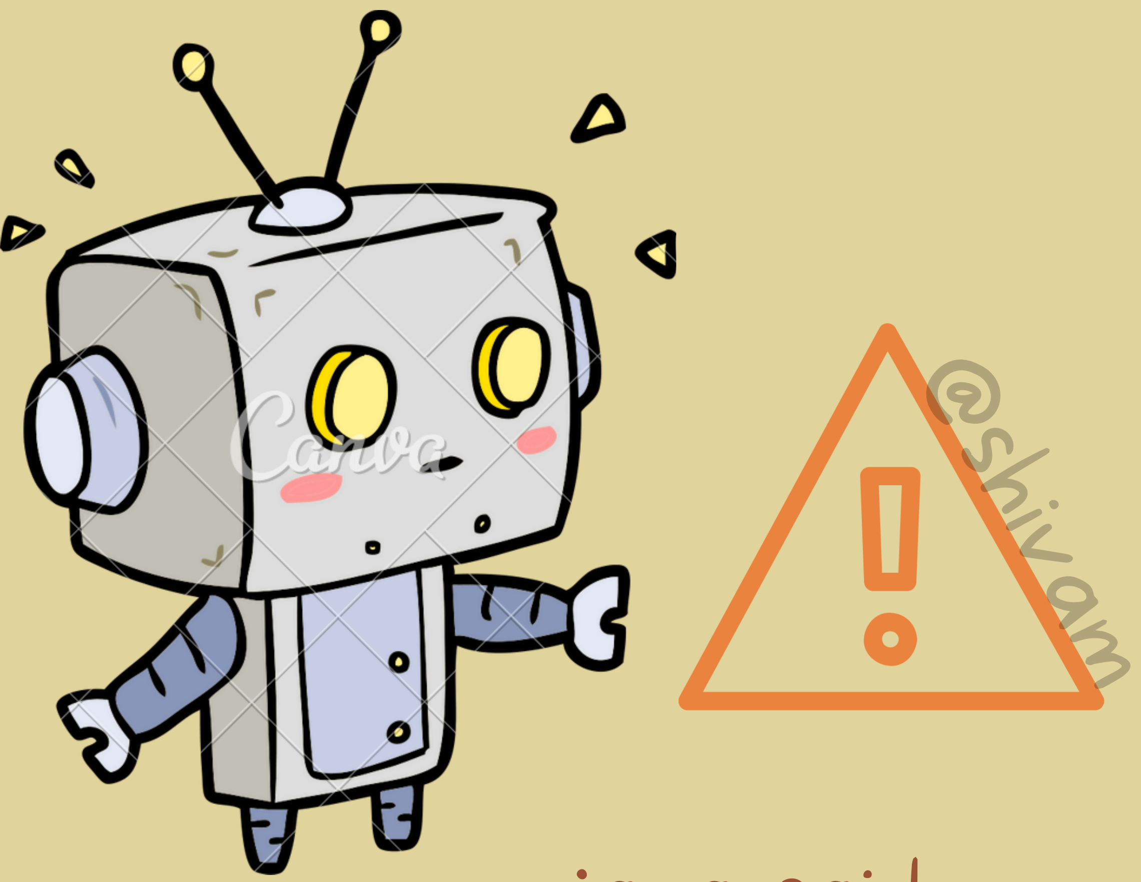
what if i pass "null" value and only sibling class is present



```
public class Sunny {

public static void main(String[] args) {
        method(null);
  }

public static void method(String string) {
          System.out.println("string");
 }

public static void method(StringBuffer sb) {
           System.out.println("stringBuffer");
 }

}
```

null is valid for both !

and they are sibling class



java said ----

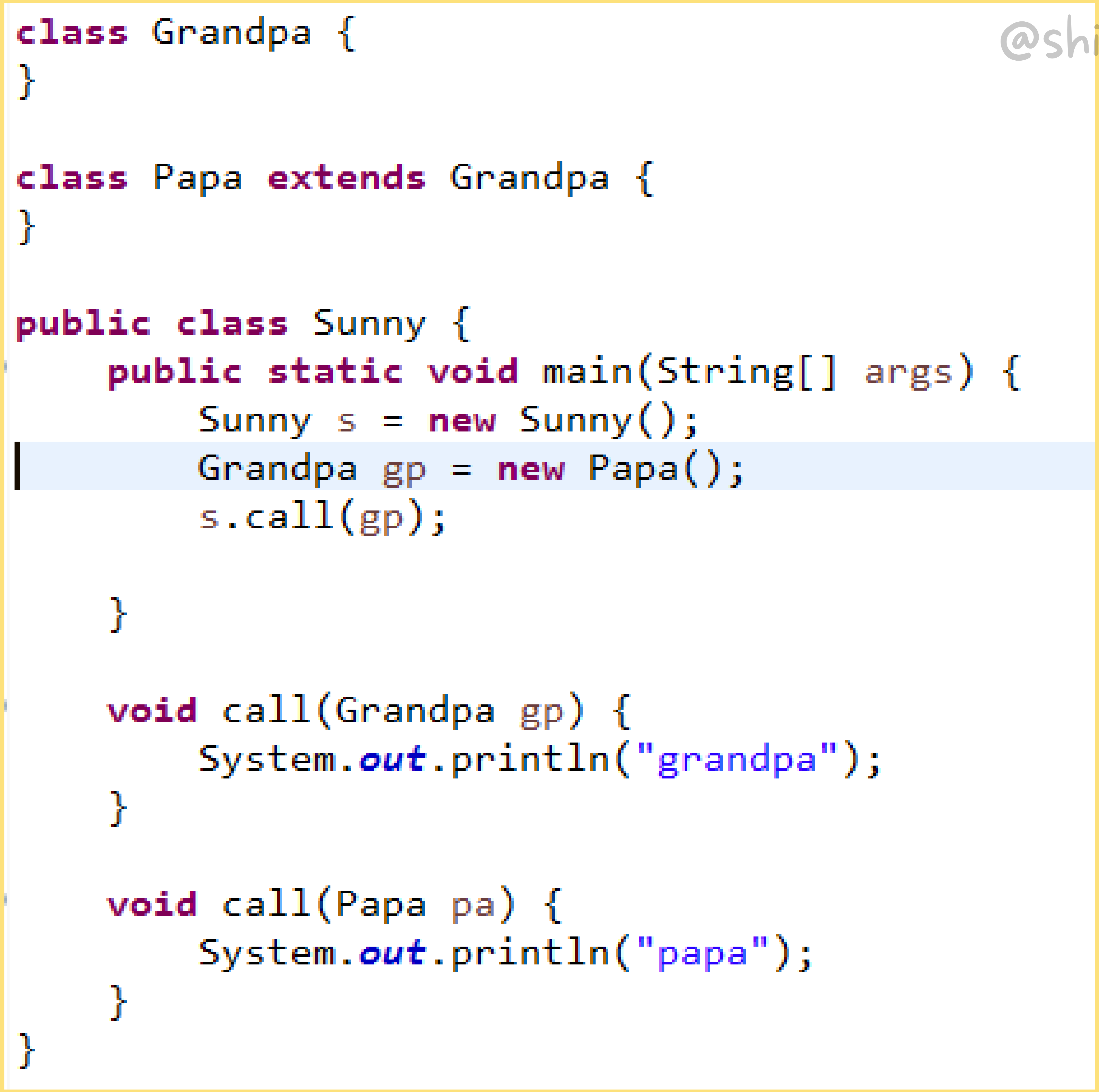
The method method(String) is ambiguous

output - error

case 7

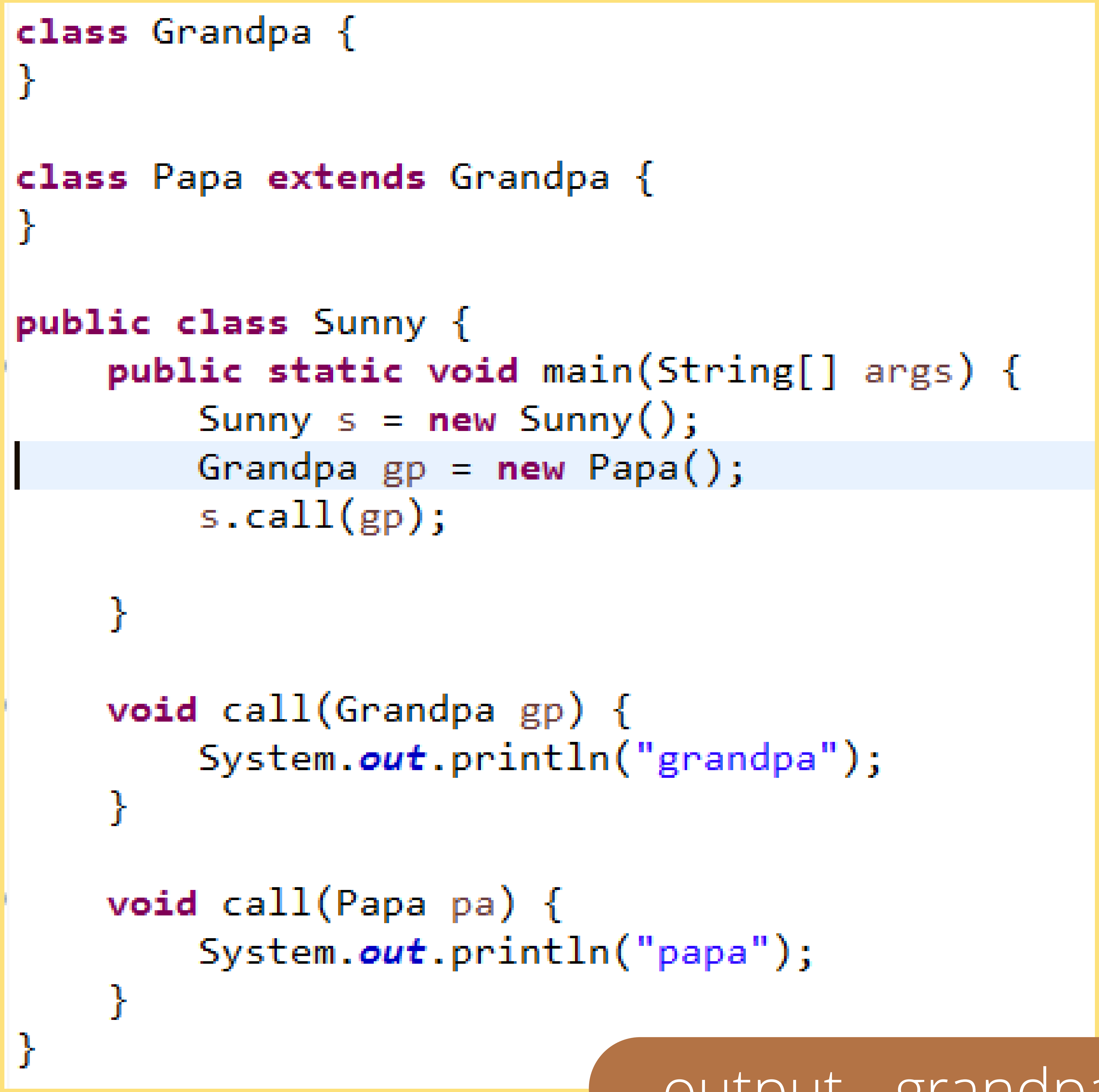
Which will get called?

gp is holding object of "papa" class but reference type is GrandPa class



```
class Grandpa {
}

class Papa extends Grandpa {
}

public class Sunny {
    public static void main(String[] args) {
        Sunny s = new Sunny();
        Grandpa gp = new Papa();
        s.call(gp);

    }

    void call(Grandpa gp) {
        System.out.println("grandpa");
    }

    void call(Papa pa) {
        System.out.println("papa");
    }
}
```

In overloading method resolution is always based on <u>reference type</u> and runtime object won't play any role in overloading.

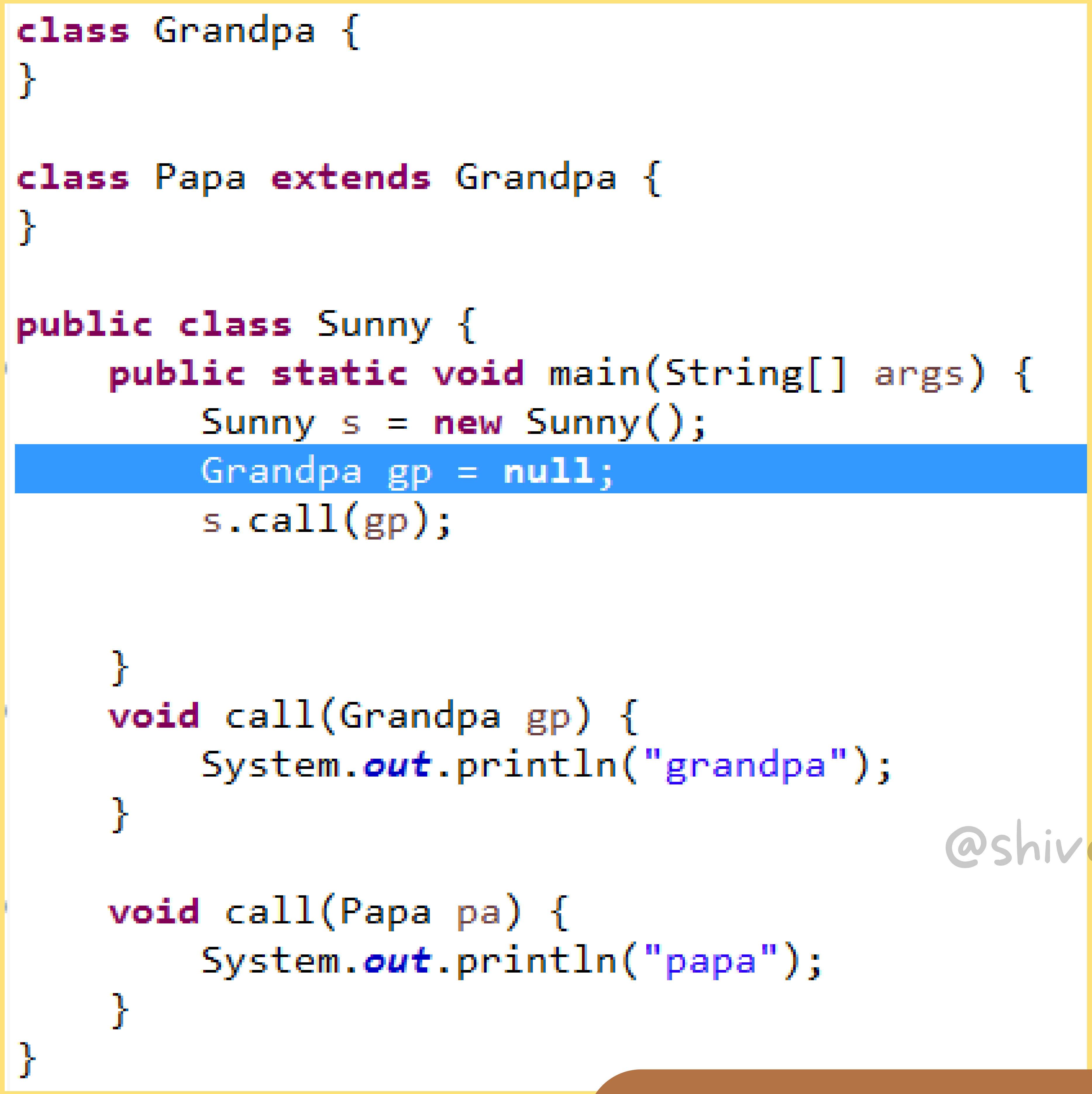


```
class Grandpa {
}

class Papa extends Grandpa {
}

public class Sunny {
    public static void main(String[] args) {
        Sunny s = new Sunny();
        Grandpa gp = new Papa();
        s.call(gp);

    }

    void call(Grandpa gp) {
        System.out.println("grandpa");
    }

    void call(Papa pa) {
        System.out.println("papa");
    }
}
```

output - grandpa

@shivam

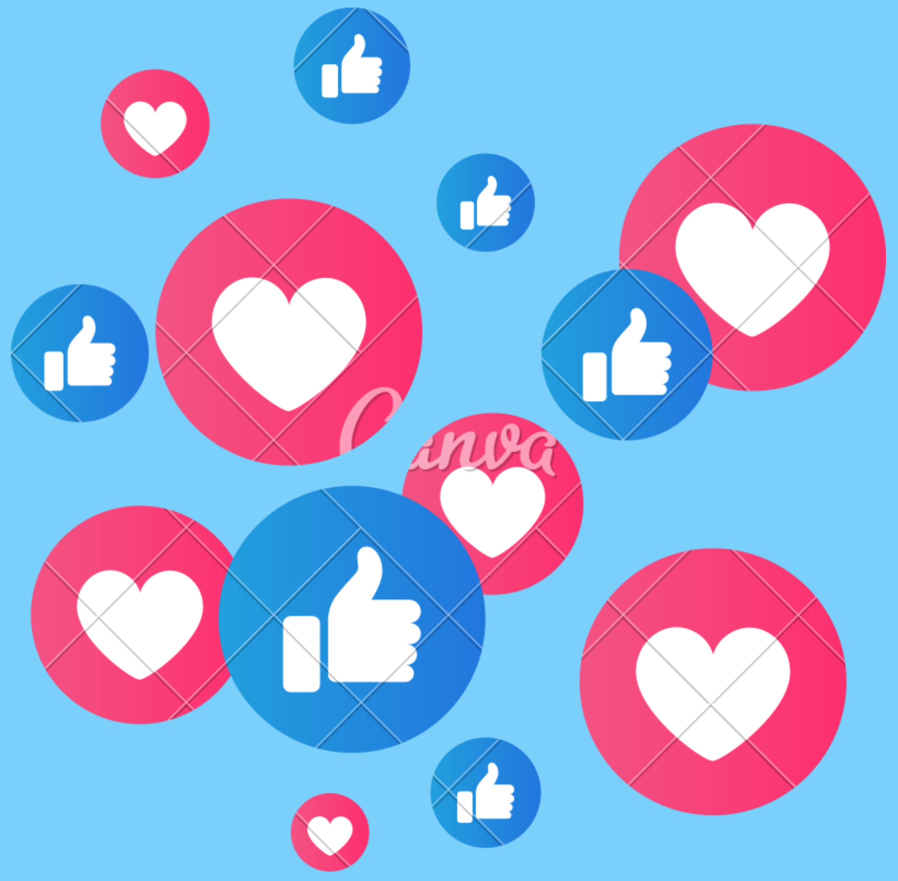
even if i place
null but
reference type is
"GrandPa"
it will call
"GrandPa" method
<u>Runtime object won't play</u>
<u>anyrole</u>



```
class Grandpa {
}

class Papa extends Grandpa {
}

public class Sunny {
    public static void main(String[] args) {
        Sunny s = new Sunny();
        Grandpa gp = null;
        s.call(gp);


    }
    void call(Grandpa gp) {
        System.out.println("grandpa");
    }

    void call(Papa pa) {
        System.out.println("papa");
    }
}
```

output - grandpa

THANK YOU



Follow for such content

@shivam