~~super keyword~~

1.) ~~super key~~

polymorphysim in java.

~~Method overloading~~

Polymorphysim

- **Compile time (static)**
  - → Method overloading
  - → Handle by compiler
  - → Same name
  - → Same class
  - → Different Arguments
    - → No. of Arguments
    - → Seq. of Arguments
    - → Types of Arguments
- **Runtime (Dynamic)**
  - → Method overriding
  - → Handle by JVM
  - → Same name
  - → Diff. class
  - → Same Arguments
    - → No. of Arguments
    - → Seq. of "
    - → Types of "
  - → Inheritance (IS-A) relationship

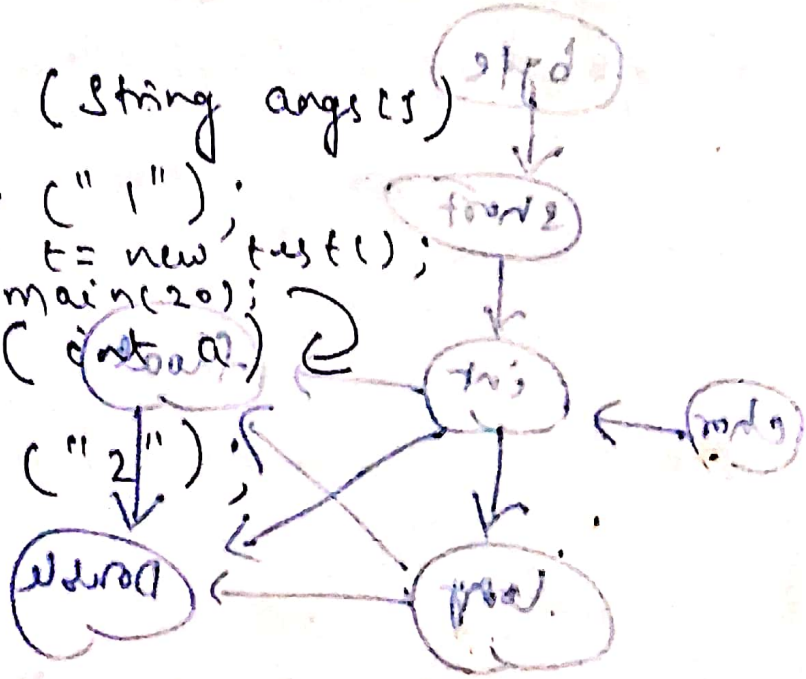
Method overloading.

```
class test
{
  void show()
  {
    s.o.p("1");
  }
  void show(int a)
  {
    s.o.p("2");
  }
  p s v m.(String args[])
  {
    test t2 = new test();
    t.show();
    t.show(3);
  }
}
```

Q) Can we overload java main() meth

```
class test
{   void show(int a)
    { s.o.p("1");
    }
    void show(String a)
    { s.o.p("2");
    }
p.s.v.m(String args[])
    { test t= new test();
      t.show(10);
    }
p.s.v.m(int a)
    { s.o.p("3");
    }
}
```

```
class test
{
    p.s.v.m(String args[])
    { s.o.p("1");
      test t= new test();
      t.main(20);
    }
p.s.v.m(int a)
    { s.o.p("2");
    }
}
```



Automatic Promotion

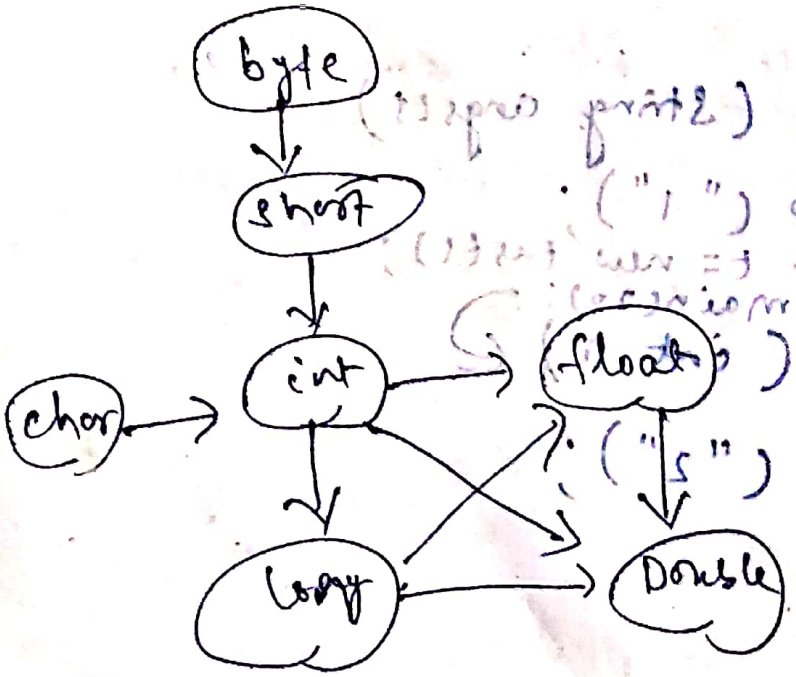
One type is promoted to another type implicitly if no matching data type is found.

e.g.

```
class test
{
void show(int a)
 { S.o.p ("int method");
 }
void show (String a)
 {
  S.o.p("String method");
 }
P.S.v.m (String args[])
{
 test t= new test();
 t.show (20);
 t.show ('a');
}
```

→ character.

↓ no match found.
Automatically converted to integer.



Method overriding:

```
class test
{
    void show()
    {
        s.o.p("1");
    }
}
class xyz extends test
{
    void show()
    {
        s.o.p("2");
    }
    p.s.v.m(String args[])
    {
        test t = new test()
        t.show();
        xyz x = new xyz();
        x.show();
    }
}
```

Q) Do overriding method must have same Return type?? Co-variant return type.

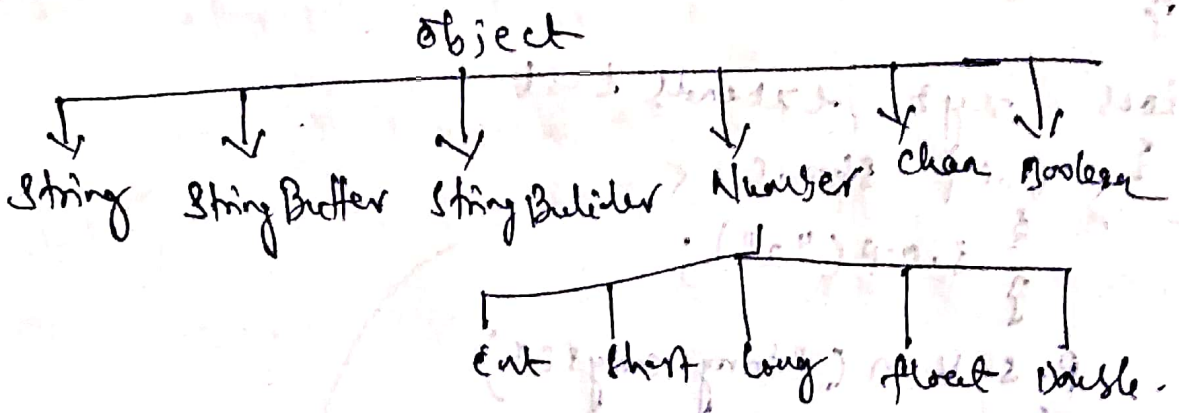
```
class test
{
    (void) show()
    {
        s.o.p("1");
    }
}
```

Object

```
class xyz extends test
{
    (void) show()
    {
        s.o.p("2");
    }
}
```

String

*) If Parent class Return type is parent datatype then we can override it in child class with child data type.

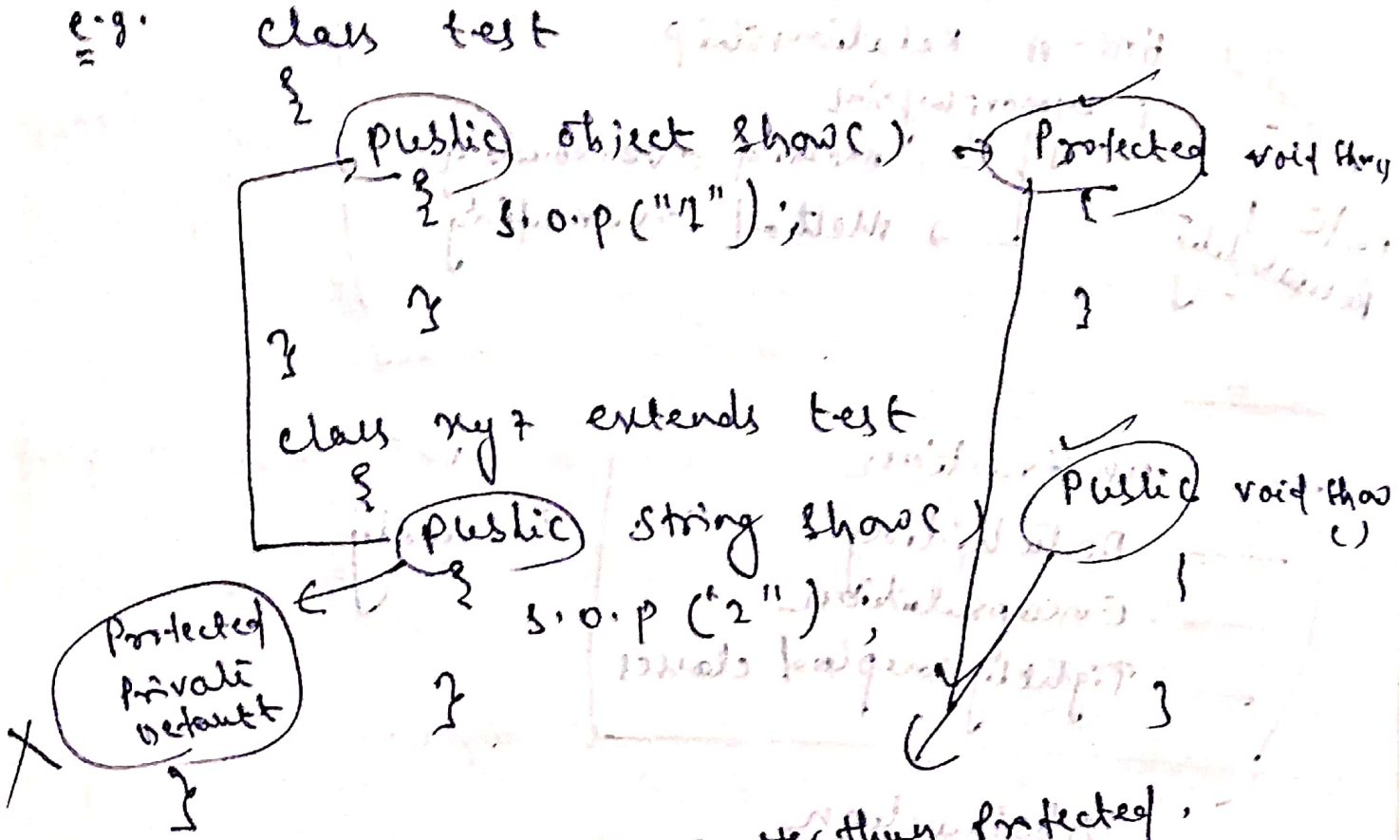



```
class test
{
    Object show()
    {
        S.o.p("1");
        return null;
    }
}
class xyz extends test
{
    String show()
    {
        S.o.p("2");
        return null;
    }
    P.S.V.M()
    {
        test t = new test();
        t.show();
        xyz oy = new xyz();
        oy.show();
    }
}
```

Q2) Access Modifiere in Overriding

*) parent class Access modifier should be same or less than child class access modifier.

e.g.



Q3)