Working with IS-A and HAS-A Relation :

* While working with IS-A and HAS-A, first of all we need to verify what kind of relation will exist between two classes.
* IS-A relation always describes sub type like Student IS-A Person, Here Student IS-A Person type.
* If the relation is HAS-A relation then we need to verify that in between two object we have **tightly coupled** relation OR **loosely coupled relation**
* If we have tightly coupled relation then we should use **Composition** where as if we have loosly coupled relation then we should use Aggregation.

Description of System.out.println() :

* System class contains a predefined class called java.io.PrintStream.

```
public final class System
{
    public static final PrintStream out = null;   [Static Blank final field]

    static
    {
          out = new PrintStream();
    }

}
```

System.out.println();

Prepare :
out = null

class Initialization
static variable + static block, both are having same priority
a) out = null
b) static block will be executed

out = PrintStream()
Object

Polymorphism :

Poly = Many
Morphism = Forms

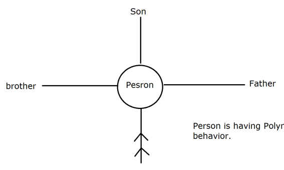
* Polymorphism is a Greek word whose meaning is **"Same Object Having different Behavior"**.

```
void person(Walking)
void person(Running)
void person(Sleeping)
void person(Riding)
```

* In our real life, a person can perform so many task (as shown above), in the same way in our programming languages a method or a constructor can perform so many task to represent polymorphic behavior.

```
class Addition
{
  public void add(int x, int y)
  {
  }
  public void add(int x, int y, int z)
  {
  }
  public void add(double x, double y)
  {
  }
}
```

Here add() method is showing polymorphic behavior because one add() method performs addition of two int, another one addition of 3 integers and one add() method performs addition of two double values these is called Polymorphic behavior.



Person is having Polymorphic behavior.