<https://www.codejava.net/java-core/the-java-language/12-rules-of-overriding-in-java-you-should-know>

**Overriding** refers to the ability of a subclass to re-implement an instance method inherited from a superclass.

Rules:

1) Method name must be same

2) Signature should be same-

No of parameters, sequence of parameter, type of parameters should be same.

3) Access specifier:

**Only inherited methods can be overridden.**

Can be same or broader type

Super: child

# If both classes are in same package-

* Public: public
* Protected: public, protected
* Default: public, protected, default
* Private: can’t be overridden

# If both classes are in different package

* Private: can’t be overridden
* Public: public
* Protected: public ,protected
* Default: NA

4)Return type

**The overriding method must have same return type (or subtype).**

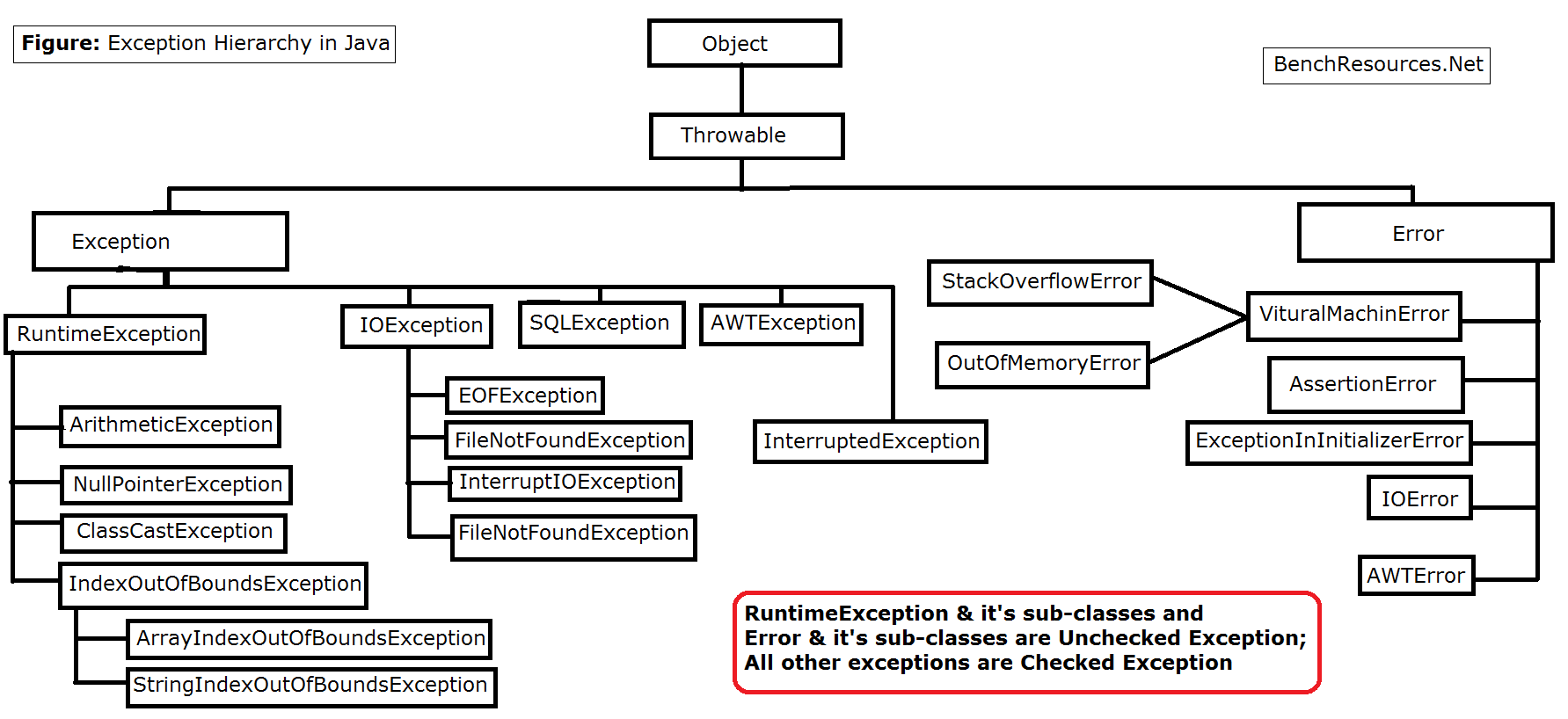
**Primitive-same**

**Covariant-same or subtype**

**5)Exception**

**The overriding method must not throw new or broader checked exceptions.**

* Unchecked: Not mandatory to handle
* Checked:
* **Same type or subtype**
* **It can not throw new exception**
* **But can throw unchecked exception**



6) **Private**,**Final and static methods cannot be overridden.**

**Static: Static methods are class level**

**Private: class level access**

**Final: keyword**

**7)** **:Use the super keyword to invoke the overridden method from a subclass.**

public class Dog extends Animal {

    protected void move() {

        super.move();   // Animal movement

        // Dog-specific moving code...

    }

}

**Rule #8:Constructors cannot be overridden.**

Because constructors are not methods and a subclass’ constructor cannot have same name as a superclass’ one, so there’s nothing relates between constructors and overriding.

**package** accessSpecifier;

**public** **class** Main {

/\* public static void gfg(String s)

{

System.out.println("String");

} \*/

**public** **static** **void** gfg(Object o)

{

System.***out***.println("Object");

}

**public** **static** **void** gfg(Throwable i)

{

System.***out***.println("throwable");

}

**public** **static** **void** gfg(Exception i)

{

System.***out***.println("exception");

}

**public** **static** **void** main(String args[])

{

*gfg*(**null**);

}

}

Output: exception

Case1) In the same level of hierarchy it will go to subclass.

Case2) In different hierarchy it will give compile error :ambiguity error