

**Lab Assignments – IV**  
**MCA Semester III**  
**CG and Java Lab (CS3307)**

1. Create a class named *Parent*, that must have a non-parameterized constructor. Now implement a method named *proc1*, that takes no parameters. Create a subclass *Child* overriding the method. Next, create an object of *Child* assigned to a *Parent* variable in main method, and invoke *proc1* from the object. Check which method is being invoked. Next, modify the access specifier of *proc1* in *Parent* as *public*, and keep the method in *Child* as default. Is that doable? If not, note down the error you have received. Next, correct the error without changing the access specifier in *Parent* class.
2. Now remove the non-parameterized constructor of *Parent*, and put a parameterized constructor of your choice. What error will you receive if you do not modify any code in *Child*? Modify the code accordingly in *Child*. Next, additionally place static and non-static constructors in both of *Parent* and *Child*, with proper printing messages. Check and state, in which sequence the initializers of the two classes are being invoked?
3. Create another subclass of *Parent* named *Child1*. Create proper constructor within it, and it must also override *proc1*. Now, create a static factory method within *Parent* class to return either *Child* or *Child1*, or null. Invoke the factory method from main, and invoke the *proc1* of the returned object. Now, change the signature of *proc1* in *Child1*, i.e., put some parameter in *proc1* of *Child1*. It becomes method overloading for *Child1* now. Elaborate the fact. If you create *Child1* object in *Parent* variable, will you be able to invoke the modified *proc1*? What error will you receive? Typecast the returned variable to *Child1*. The error should be gone now. Why is it giving error without typecasting?
4. Modify the *Parent* class to an abstract class and make *proc1* an abstract method. Now you should receive an error in *Child1*. Why? Correct the error in *Child1*. Will the initializers in *Parent* class be invoked now, if an object of *Child* or *Child1* is created? Experiment with the fact.
5. Place the *Child1* in another package *pkg*. Create 4 variables in *Parent*, each with different access specifiers. Access these variables from *Child* and from *Child1*. Moreover, while accessing in *Child1*, access the variables in two different ways, firstly via direct inheritance just as if they are *Child1*'s own members, and secondly, by creating an object of *Parent* in *Child1* and accessing through that. Create two more classes which are not subclass of *Parent*, one in the same package as *Parent*, and one in the different package *pkg*. Access these variables from the newly created subclasses as well. Observe and note down the access rules.