**Important top 7 Java method overriding interview programs**

**1. What is the output of the program**

package overridingPrograms;

public class A

{

private void m1()

{

System.out.println("m1-A");

}

}

public class B extends A

{

private void m1()

{

System.out.println("m1-B");

}

public static void main(String[] args)

{

B b = new B();

b.m1();

}

}

**Ans: No, Compile time error.**

**Explanation: The overriding concept is not applicable to the private method. Parent class private method is not visible in the child class. Keep in mind. Based on our requirement, we can define exactly the same private method in the child class. The code is valid but not overriding.**

**2. Will the below code compile successfully?**

package overridingPrograms;

public class A

{

public final void m1()

{

System.out.println("m1-A");

}

}

public class B extends A

{

public void m1()

{

System.out.println("m1-B");

}

public static void main(String[] args)

{

B b = new B();

b.m1();

}

}

**Ans:No, Compile-time error.**

**Explanation: If the parent class method is declared as final, you cannot override it in the child class. If you are trying to override it, you will get the compile-time error-“Overridden method is final”.**

**3. Will the below code compile successfully? If yes, what will be the output?**

package overridingPrograms;

public class A

{

protected void m1()

{

System.out.println("m1-A");

}

}

public class B extends A

{

public final void m1()

{

System.out.println("m1-B");

}

public static void main(String[] args)

{

B b = new B();

b.m1();

A a = new B();

a.m1();

}

}

Ans: Yes, the code will be successfully compiled.

**o/p**

**m1-B**

**m1-B**

**Explanation: You cannot declare the parent class method as final but the child class method can be declared as final. It is used to restrict further overriding.**

**a) When b.m1() will be executed, it will call m1() of class B because the reference variable ‘b’ is pointing to the objects of class B. So, the output will be “m1-B”.**

**b) Since ‘a’ is also pointing to the objects of class B. Therefore, the output will be “m1-B”.**

**4. Can you find out the error in the below code?**

public class P

{

static void m1()

{

System.out.println("Class P");

}

}

public class Q extends P

{

@Override

static void m1()

{

System.out.println("Class Q");

}

}

**Ans: The overriding concept is not applicable to a static method.**

**5. What will be the output of the following program?**

public class XY

{

protected Number m1(int a)

{

System.out.println("One");

return null;

}

Object m2()

{

System.out.println("Two");

return null;

}

}

public class YZ extends XY

{

protected String m2()

{

System.out.println("Three");

return null;

}

}

public class XYZ

{

public static void main(String[] args)

{

XY xy = new YZ();

xy.m1(20);

xy.m2();

}

}

**Output:**

**One**

**Three**

**6. In the below snippet code, Is m1() correctly overridden in the subclasses of class X?**

public class One

{

void m1()

{

}

}

public class Two extends One

{

@Override

protected void m1()

{

System.out.println("m1-Two");

}

}

public class Three extends Two

{

@Override

public void m1()

{

System.out.println("m1-Three");

}

}

**Ans: Yes**

**Explanation: m1() method is correctly overridden in the subclasses of class X.**

**7. What will be the output of the following program?**

package overridingPrograms;

public class One

{

void m1() throws Throwable

{

System.out.println("m1-One");

}

}

public class Two extends One

{

@Override

protected void m1() throws Exception

{

System.out.println("m1-Two");

}

}

import java.io.IOException;

public class Three extends Two

{

@Override

public final void m1() throws IOException

{

System.out.println("m1-Three");

}

}

public class MyTest

{

public static void main(String[] args) throws Throwable

{

One o = new Two();

o.m1();

Two t = new Three();

t.m1();

Three th = new Three();

th.m1();

}

}

**Output:**

**m1-Two**

**m1-Three**

**m1-Three**