Assignment

Topic:-OOPS

Name&Batch:-Ch. Anil Kumar(60R)

1. **Difference between Method Overloading and Method Overriding in terms of Inheritance**

   Method Overloading:-

   - Same method name but different parameters, within the same class.
   - Overloading does not depend on inheritance. It can happen in a single class itself, but also possible in parent-child classes.
   - It is a compile-time polymorphism

   Method Overriding:-

   - Same method name ,same parameter and same return, within the same class
   - Method Overriding occurs when a child class (subclass) defines a method with the same name, same parameters, and same return type as a method in its parent class (superclass), but provides a different implementation.
   - It is a run-time polymorphism

Example:-

```
class Parent {


   void display() {
      System.out.println("Parent class method");
   }


   // Overloaded methods
   void add(int a, int b) {
      System.out.println("Sum: " + (a + b));
   }
   void add(int a, int b, int c) {
      System.out.println("Sum: " + (a + b + c));
   }
}


class Child extends Parent {
   // Overriding the method
   @Override
```

```
    void display() {
        System.out.println("Child class method");
    }
}

public class Main {
    public static void main(String[] args) {
        Parent p = new Parent();
        p.display();
        p.add(2, 3);
        p.add(1, 2, 3);
        Parent o = new Child();
        o.display();
    }
}
```

2. **Can you inherit a private method? If not, why?**
    - No, a private method cannot be inherited.
    - Reason: Private methods are accessible only inside the class where they are declared.
    - If you declare the same method in child class, it will be treated as a new method, not an override.

Example:-

```
class Parent {
    private void secret() {
        System.out.println("This is a parent secret method.");
    }

    void callSecret() {
        secret();
    }
}

class Child extends Parent {
        void secret() {
        System.out.println("This is a child method, not inherited.");
```

```
    }
}


public class Main {
    public static void main(String[] args) {
        Child c = new Child();
        c.secret();
   c.callSecret();
    }
}
```

3. **Can a final class be inherited?**
   - No, a final class cannot be inherited**.**
   - Final is a keyword ,final means can't be modified or updated
   - The final keyword is used to prevent inheritance**.**
   - final class is cannot be inherited.
   - final method is cannot be overridden.

   Example:-

```
final class Vehicle {
    void run() {
        System.out.println("Vehicle is running");
    }
}
// class Car extends Vehicle { }


public class Main {
    public static void main(String[] args) {
        Vehicle v = new Vehicle();
        v.run();
    }
}
```

4. **What will happen if a parent class reference points to a child class object?**
   - This is called Upcasting in Java (when a parent class reference points to a child class object).
   - It's allowed in Java.
   - With upcasting, the parent reference can only access methods that are defined in the parent class.
   - But if the child class has overridden a method, then the child's version will be executed — this is called runtime polymorphism.

Example:-

```
class Parent {
    void show() {
        System.out.println("Parent show()");
    }
}

class Child extends Parent {
    @Override
    void show() {
        System.out.println("Child show()");
    }
    void onlyChild() {
        System.out.println("Only child method");
    }
}

public class Main {
    public static void main(String[] args) {
        Parent ref = new Child();
        ref.show();
        }
}
```

5. **Can you access the parent class constructor from a child class?**
   - Yes, you can call a parent class constructor from a child class using the super() keyword.
   - In fact, the parent's constructor always runs before the child's constructor.
     If you don't write super() explicitly, Java automatically adds it for you.

Example:-

```
class Parent {
  Parent() {
    System.out.println("Parent constructor called");
  }
}
```

```
class Child extends Parent {
    Child() {
        super();
        System.out.println("Child constructor called");
    }
}

public class Main {
    public static void main(String[] args) {
        Child c = new Child();
    }
}
```