Skip limit.

```
public class SkipLimit {
    public static void main(String[]
{
    List<Item> fruit = List.of(
    new Item("shoe", 10, 7.99),
    new Item("blue", 20, 26.99),
    new Item("apple", 10, 28.99),
    new Item("kimi", 10, 11.99));

List<Item> top3 = fruit.stream()
    .sorted(Comparator.comparing(
        Item::getPrice))
    .limit(3)
    .collect(Collection.toList());

System.out.println("top3" + top3);

List<Item> top3Rlest = fruit.stream
    .sorted(Comparator.comparing(
        Item::getPrice).reversed()
    .limit(3)
    .collect(Collection.toList());
System.out.println(top3Rlest);

}
}
```

Advanced Object Oriented Concepts

Q] What is Polymorphism?

→ Polymorphism is defined as "Same Code" giving "Different Behavior". Let's look at an example. Let's define an Animal class with a method shout.

```
public class Animal {
    public String shout() {
        return "Don't Know";
    }
}
```

Let's create two new sub classes of Animal overriding the existing shout method in Animal.

```
class Cat extends Animal {
    public String shout() {
        return "Meow Meow";
    }
}
class Dog extends Animal {
    public String shout() {
        return "Bow";
    }
    public void run() {
    }
}
```

Polymorphism is one of the core concepts of object-oriented programming that describes situation in which something occurs in several different forms.

In computer science, polymorphism describes the concept that you can access objects of different types through the same interface. Each type can provide its own independent implementation of this interface.

Types of Polymorphism

1) Static or Compile-time
2) dynamic.

1) Static polymorphism -

→ Java allows you to implement multiple method with in the same class that use the same name, But Java uses a different set of parameters called method overloading. It represent static form of polymorphism.

The parameters sets have to differ in at least one of the following three criteria:

1) They need to have different number of parameters. one method accepting 2 & another one accepting 2 parameters.

2) In types of parameters need to be different, with one method accepts a string & another one accepting a long.

3) They need to expect in parameters in a different order. For example, one method accepts a string & a long, & another one accepts a long & a string, this kind of overloading is not advisable because it makes the API difficult to understand.

2] ① Dynamic Polymorphism

This form of polymorphism does allow the compiler to determine the executed method. The JVM need to do that at runtime.

1] within an inheritance hierarchy, a subclass can override a method of it's superclass, enabling the developer of the subclass to ~~to~~ customize or completely replace the behavior of the method.

q) what is the Use of instance Operator in Java?

-> Instance Operator check if an object is of a particular type. Let us consider the following class & interface declaration.

The Java "instance" Operator is used to test whether the object is an instance of the specified type (class or subclass or interface). It is also known as type comparison operator because it compares the instance with type. It return either true or false. if we apply this Operator with any variable that has null value returns false.

q) what is Coupling?

-> Coupling is a measure of how much a class is dependent on other classes. There should minimal dependencies between classes, so, we should always aim for low Coupling between classes.

Coupling is nothing but the dependency of one class on the other. if one object in a code uses the other in the program, it is called base Coupling in java.

In Coupling, two classes or objects collaborate of work with each other to complete a pre-defined task. It simply means that one element requires another element to complete a function. It is known as collaboration when one class calls in logic of the other class.

Types of Coupling

1) Loose Coupling
2) Tight Coupling

	Loose	Tight
1]	object are independ of each other	one object is depended on the other object to complete a task
2]	Better testability	Testability is not as great as loose Coupling in Java
3]	Asynchronous Communication	Synchronous Communi -cation.
4]	Less Coordination. Swapping code betn two classes in not easy.	Provider better Coordination. You can easily swap code betn two objects.
5]	No Concept of interface	Follows GOF principles to interface.
6]	Less information flow	More information flow.
7]	Highly changeables	It does not have a change capability.

Q) what is Cohesion?

→

Cohesion is measure of how related the responsibilities of a class are. A class must be highly cohesive i.e. it's responsibilities (methods) should be highly related to one another.

In Java Cohesion is defined as used to perform the specialized tasks (single task) instead of multiple tasks with a single Java class. If a class is dealing with high Cohesion then it is said to targeted towards a single specific purpose, rather than performing different tasks at a time.

Type of Cohesion

1) Low Cohesion
2) High Cohesion

1) Low Cohesion

→ when class is designed to perform many different tasks instead of focusing on any specific task then that class is called "low Cohesive class".
This kind of approach is in bad programming design approach.
It required a lot of modification for small change.

2] High Cohesion -

when class is designed to perform any specific task then that class is called as "High Cohesive class". This kind of approach is good programming design approach. It can easily maintain & less modifiable.

Q] what is An Anonymous class ?

-> In Java, a class can contain another class known as nested class. It's possible to create a nested class without giving any name

A nested class ~~must be~~ that don't have any name is know as anonymous class.

An anonymous class must be defined inside another class hence, it is also known as an anonymous inner class.