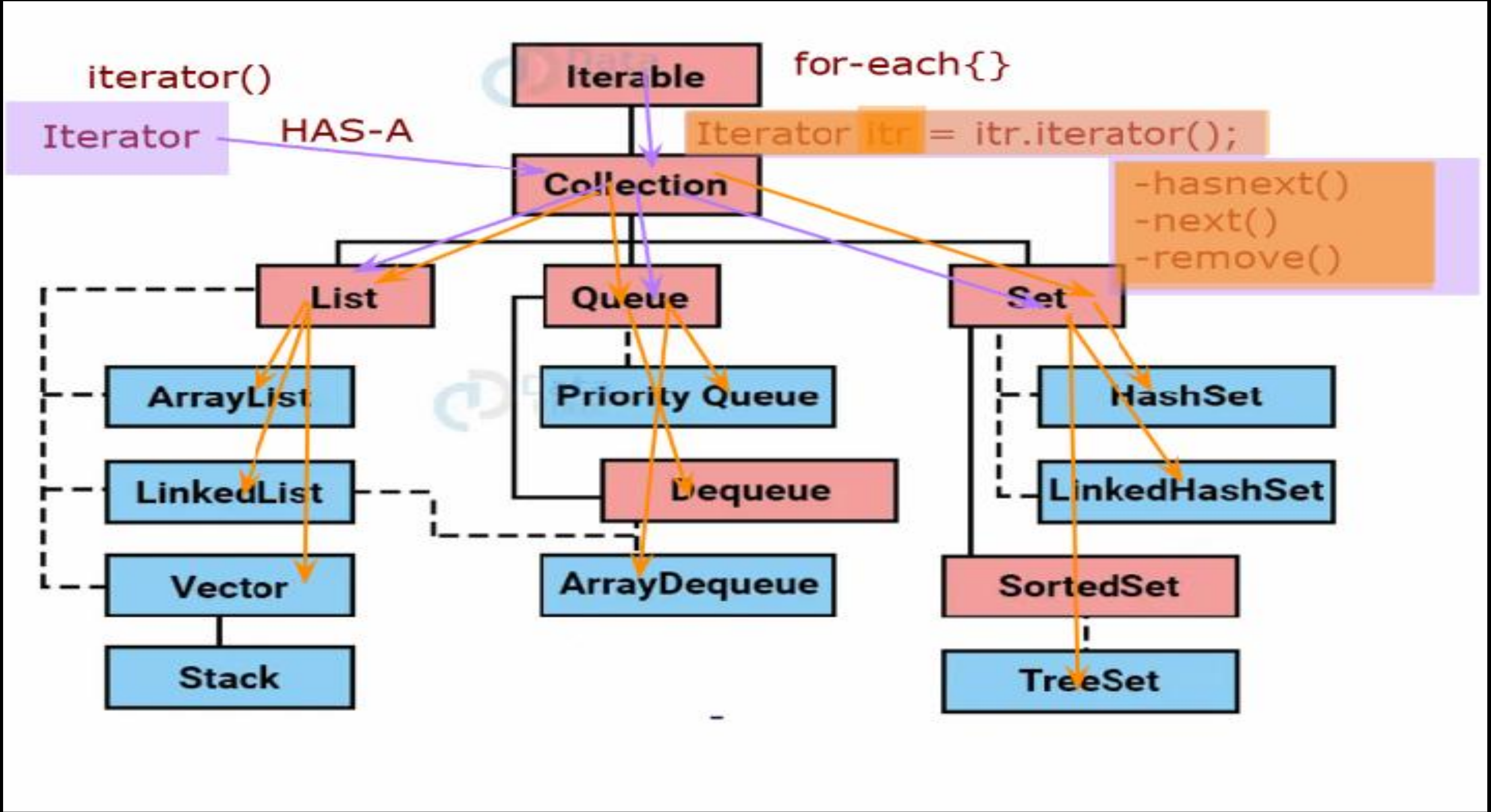
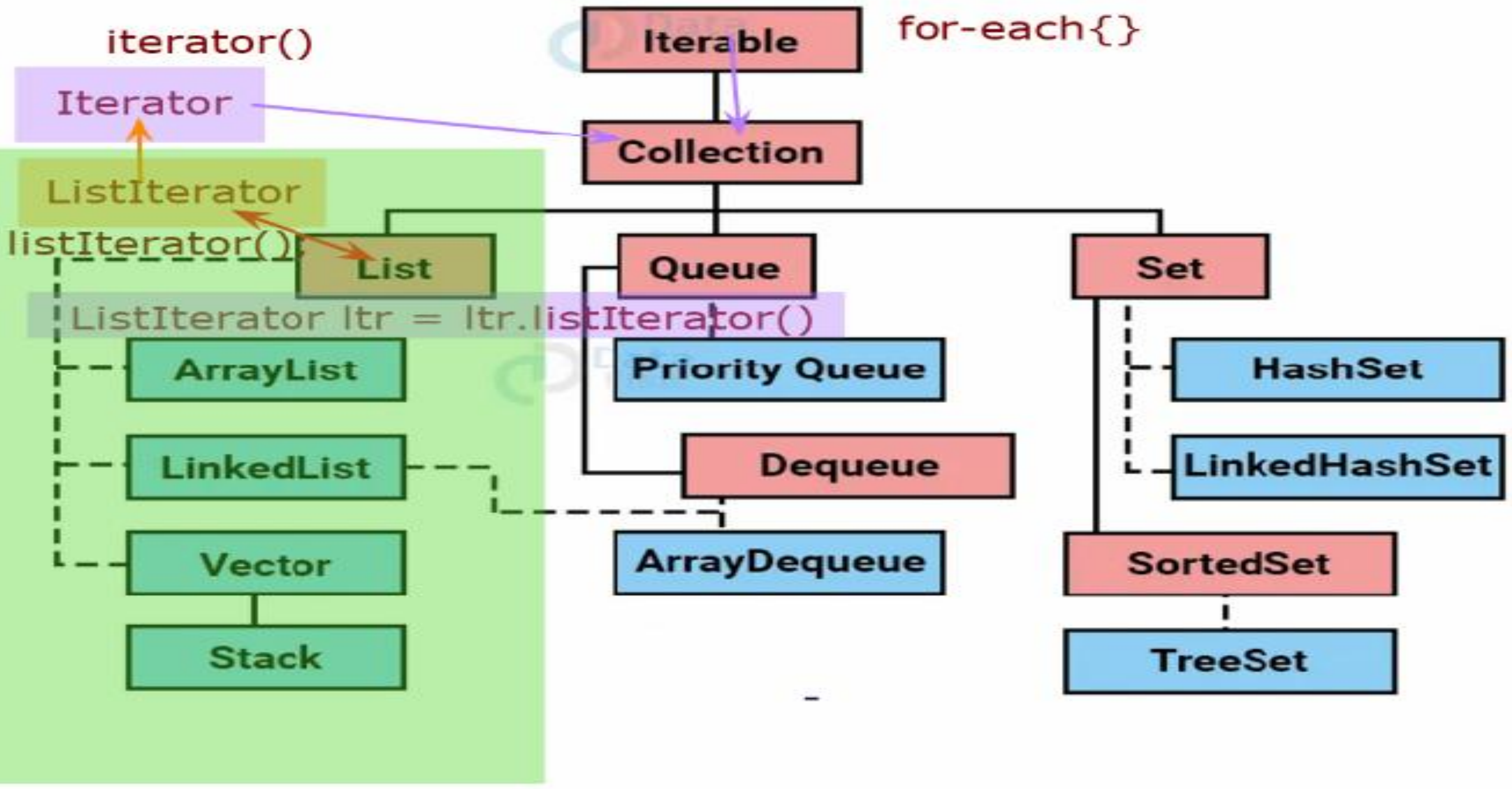
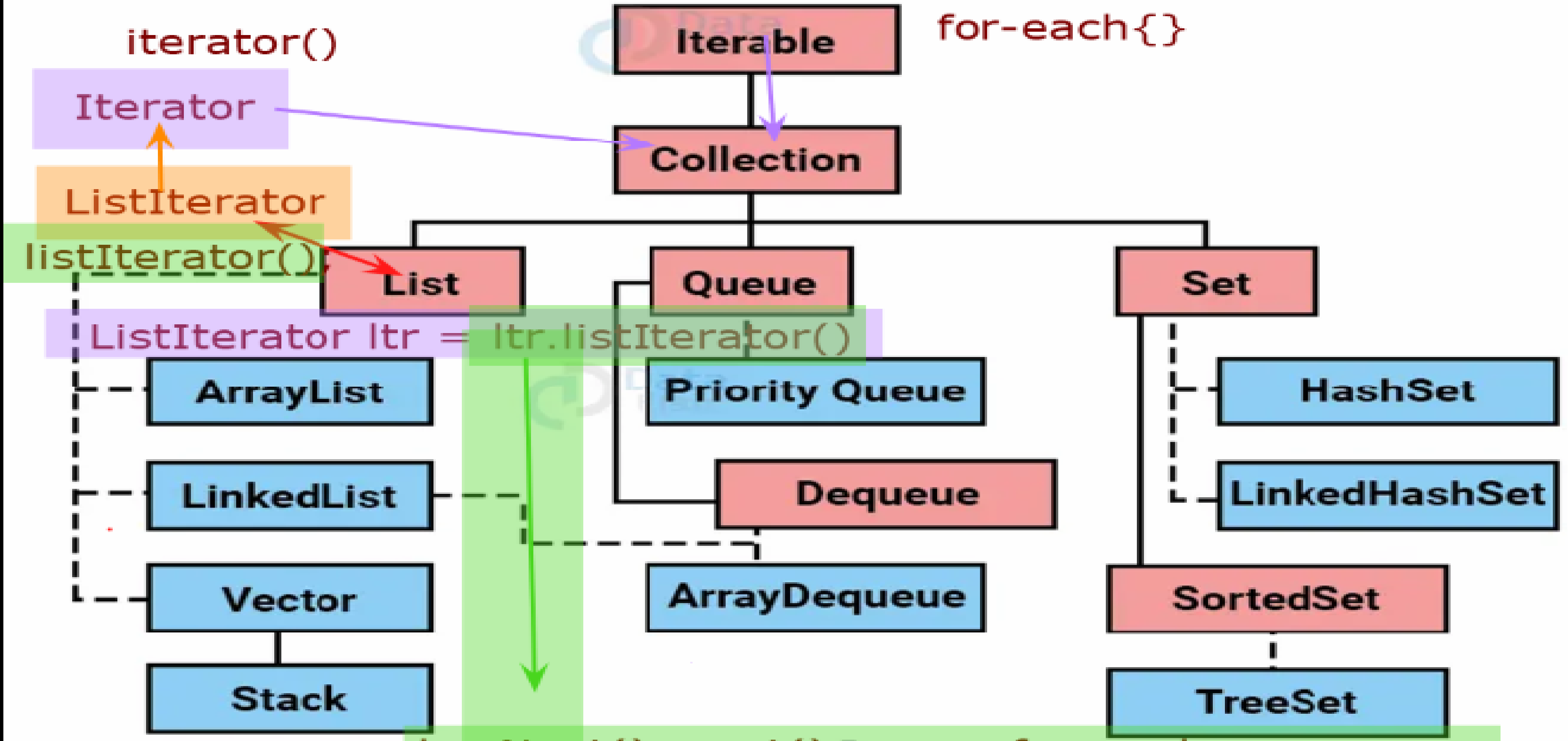
Object Oriented Programming with Java (OOPJ)

Session 5: Arrays

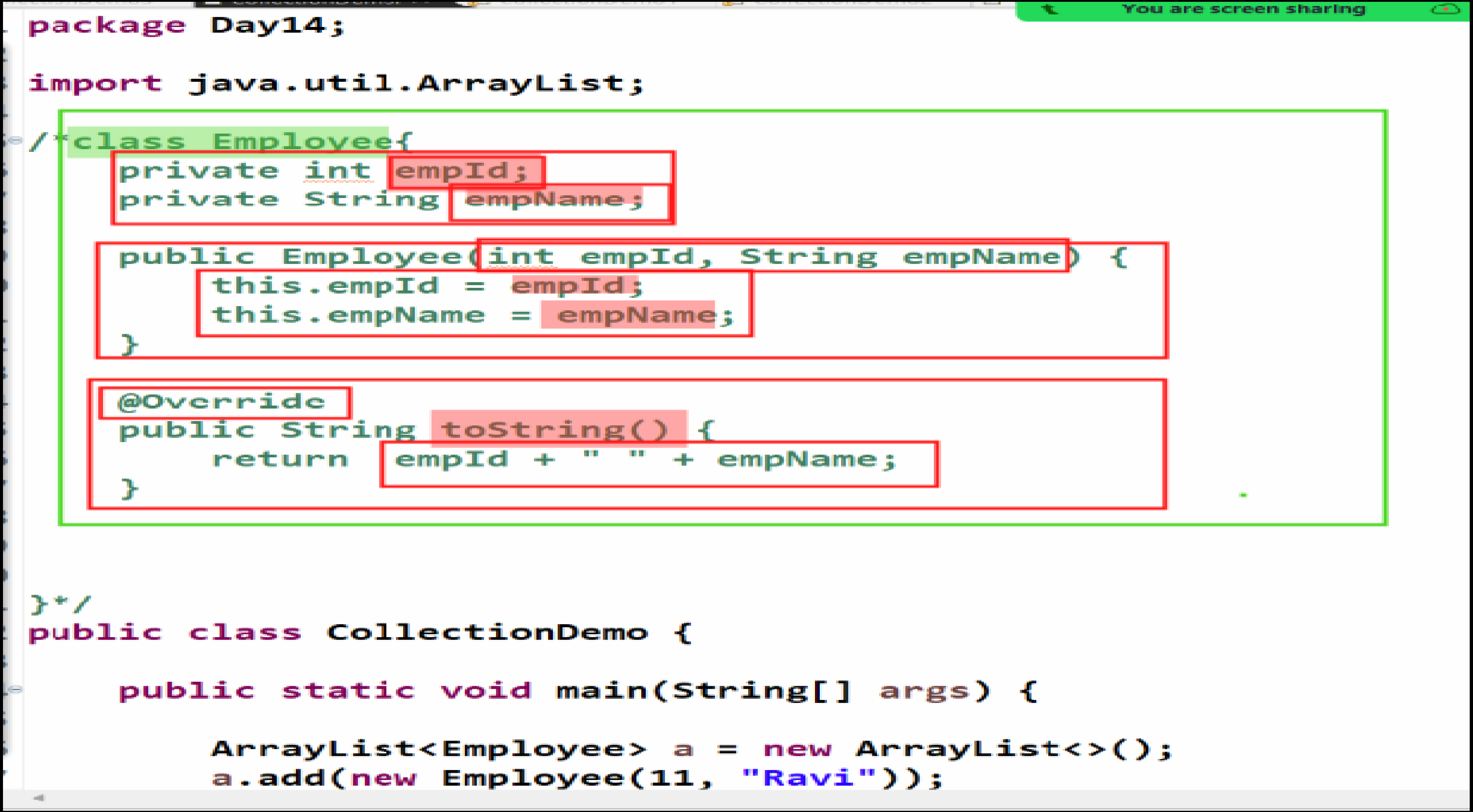
Kiran Waghmare


iterator()
Iterable
for-each{}
Iterator
HAS-A
Iterator itr = itr.iterator();
Collection
-hasnext()
-next()
-remove()
List
Queue
Set
ArrayList
Priority Queue
HashSet
LinkedList
Dequeue
LinkedHashSet
Vector
ArrayDequeue
SortedSet
Stack
TreeSet


iterator()
Iterator
ListIterator
listIterator()
Iterable
for-each{}
Collection
List
Queue
Set
ListIterator ltr = ltr.listIterator()
ArrayList
Priority Queue
HashSet
LinkedList
Dequeue
LinkedHashSet
Vector
ArrayDequeue
SortedSet
Stack
TreeSet

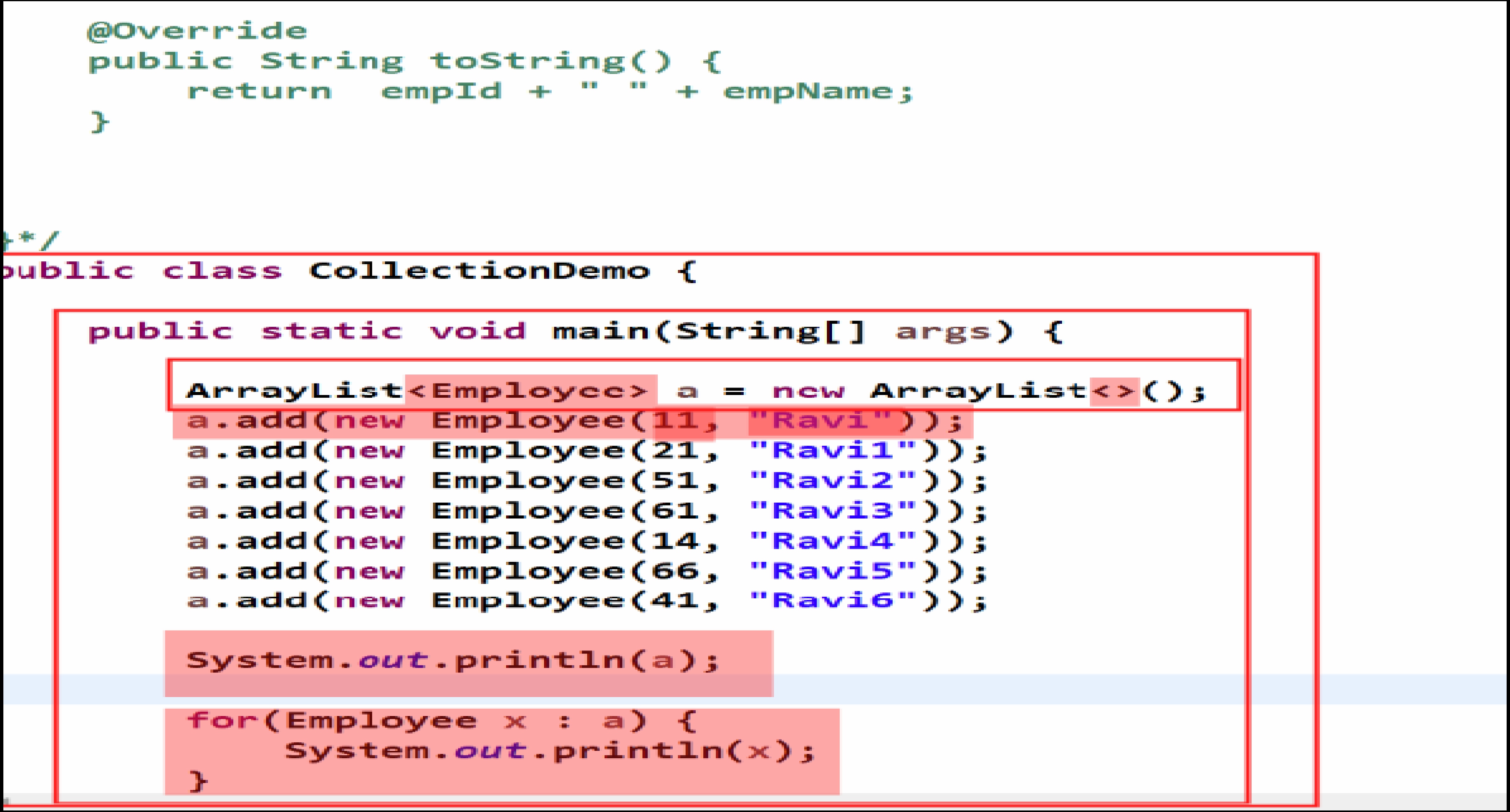


-hasNext(), next(): move forward
-hasPrevious(), previous(): move backward
-add()
-set()

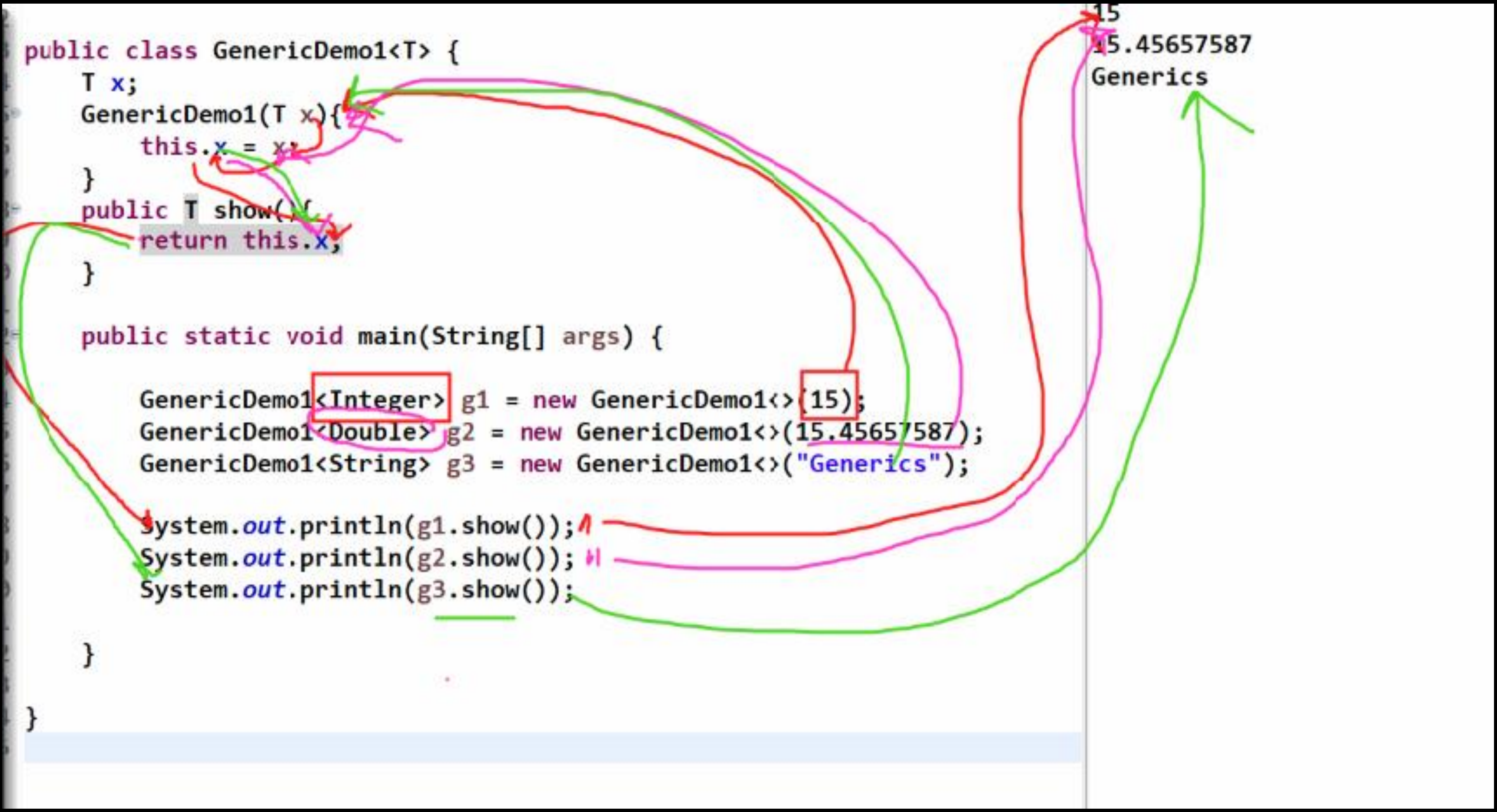


You are screen sharing

```
package Day14;

import java.util.ArrayList;

/*class Employee{
    private int empId;
    private String empName;

    public Employee(int empId, String empName) {
        this.empId = empId;
        this.empName = empName;
    }

    @Override
    public String toString() {
        return empId + " " + empName;
    }


}*/
public class CollectionDemo {

    public static void main(String[] args) {

        ArrayList<Employee> a = new ArrayList<>();
        a.add(new Employee(11, "Ravi"));
```

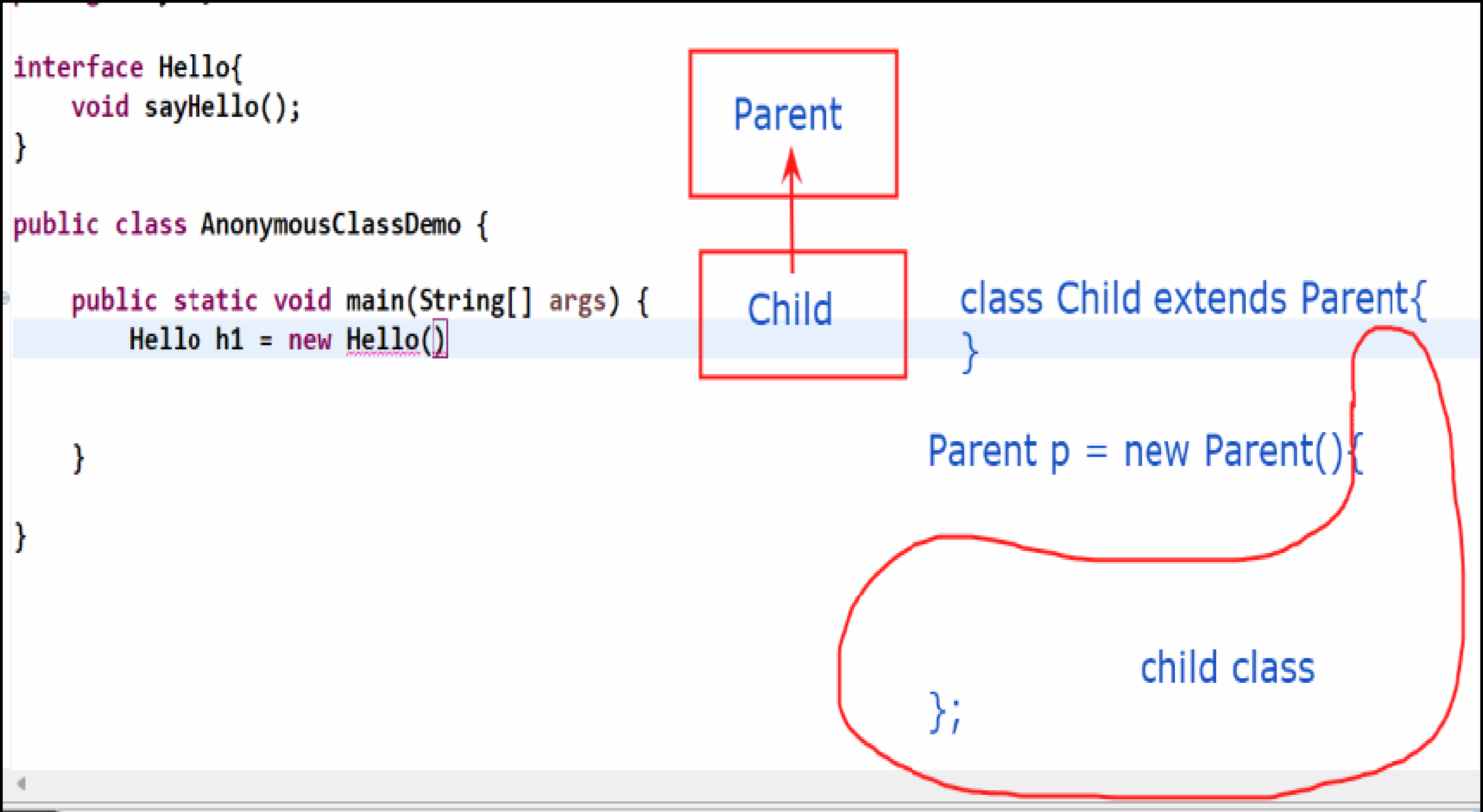


```
    @Override
    public String toString() {
        return  empId + " " + empName;
    }


}*/
public class CollectionDemo {

    public static void main(String[] args) {

        ArrayList<Employee> a = new ArrayList<>();
        a.add(new Employee(11, "Ravi"));
        a.add(new Employee(21, "Ravi1"));
        a.add(new Employee(51, "Ravi2"));
        a.add(new Employee(61, "Ravi3"));
        a.add(new Employee(14, "Ravi4"));
        a.add(new Employee(66, "Ravi5"));
        a.add(new Employee(41, "Ravi6"));

        System.out.println(a);

        for(Employee x : a) {
            System.out.println(x);
        }
```



```java
public class GenericDemo1<T> {
    T x;
    GenericDemo1(T x){
        this.x = x;
    }
    public T show(){
        return this.x;
    }

    public static void main(String[] args) {

        GenericDemo1<Integer> g1 = new GenericDemo1<>(15);
        GenericDemo1<Double> g2 = new GenericDemo1<>(15.45657587);
        GenericDemo1<String> g3 = new GenericDemo1<>("Generics");

        System.out.println(g1.show());
        System.out.println(g2.show());
        System.out.println(g3.show());

    }

}
```

```
15
15.45657587
Generics
```



```
interface Hello{
    void sayHello();
}

public class AnonymousClassDemo {

    public static void main(String[] args) {
        Hello h1 = new Hello()


    }

}
```

```
class Child extends Parent{
}

Parent p = new Parent(){


                child class
};
```

```
class Outer {
    class Inner {
        void show() {
System.out.println("Inner class method"); }
    }
}

public class Test {
    public static void main(String[] args) {
        Outer outer = new Outer();
        Outer.Inner inner = outer.new Inner();
        inner.show();
    }
}
```

```java
class Outer {
    static class Inner {
        void show() { System.out.println("Static Nested Class"); }
    }
}

public class Test {
    public static void main(String[] args) {
        Outer.Inner inner = new Outer.Inner();
        inner.show();
    }
}
```

```
class Outer {
    void outerMethod() {
        class LocalInner {
            void display() { System.out.println("Local Inner Class"); }
        }
        LocalInner obj = new LocalInner();
        obj.display();
    }
}
```

```
abstract class A {
    abstract void sound();
}

public class Test {
    public static void main(String[] args) {
        A obj = new A() { // Anonymous Inner Class
            void sound() {
                    System.out.println("Roar!"); }
        };
        obj.sound();
    }
}
```

```
class Outer {
    int x = 10;
    static class Inner {
        void display() {
            System.out.println(x);
            System.out.println("Static nested class");
        }
    }
}
```

```
class Outer {
    class Inner {
         static void show() {} }

    static class StaticInner {
        static void show() { System.out.println("Allowed in static
nested class"); }
    }
}
```