Why?

```
package com.arraylist;
import java.util.ArrayList;
import java.util.Collections;
public class ArrayListDemo2 {
	public static void main(String[] args) {

		ArrayList<Integer> al = new ArrayList<Integer>(); // generics
		al.add(10);
		al.add(20);
		al.add(15);
		al.add(12);
		al.add(3);
		System.out.println("before sorting>>" + al);
		Collections.sort(al);
		System.out.println("after sorting>>" + al);

	}
}
```

Output-

```
before sorting>>[10, 20, 15, 12, 3]
after sorting>>[3, 10, 12, 15, 20]
```

Suppose I have one employee class that containing following items

```
package com.comp;
public class Employee{
    int id;
    String name;
    int salary;

    public Employee(int id, String name, int salary)
{
        this.id = id;
        this.name = name;
        this.salary = salary;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getSalary() {
        return salary;
    }
```

```
    public void setSalary(int salary) {
        this.salary = salary;
    }

    @Override
    public String toString() {
        return "Employee [id=" + id + ", name=" +
name + ", salary=" + salary + "]";
    }

}
```

Test.Java

```
package com.sort;


import java.util.ArrayList;
import java.util.Collections;


public class Test {


    public static void main(String[] args) {


        ArrayList<Employee> al = new ArrayList<Employee>();
        al.add(new Employee(101, "ram", 9000));
        al.add(new Employee(102, "ashok", 3000));
        al.add(new Employee(103, "ajay", 8000));


        Collections.sort(al);


    }
}
```

Here, I am getting error at collection.sort(al);

Now to resolve this issue, need to implement comparable<Employee> into Employee class.

Comparable interface-

1. This interface is present in java.lang package.
2. It contain the only one method compareTo()

   Public int compareTo(Object obj)

3. Example- obj1.compareTo(obj2)

   return 1 if obj1 is greater than obj2

   return -1 if obj1 is less than obj2

   return 0 if obj1 & obj2 are equal.

Example-1

```
package com.sort;

public class Employee implements Comparable<Employee> {

	int id;
	String name;
	int salary; // custom objects

	public Employee(int id, String name, int salary) {
		super();
		this.id = id;
		this.name = name;
		this.salary = salary;
	}

	public int getId() {
		return id;
	}
```

```
	public void setId(int id) {
		this.id = id;
	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}

	public int getSalary() {
		return salary;
	}

	public void setSalary(int salary) {
		this.salary = salary;
	}

	// sort data by using salary, comparsion, id
	@Override
	public int compareTo(Employee employee) {
		//equal must return 0, > return 1, < return -1
		if (salary == employee.salary)
			return 0;
		else if (salary > employee.salary)
			return 1;
		else
			return -1;
	}

	@Override
	public String toString() {
		return "Employee [id=" + id + ", name=" + name + ",
salary=" + salary + "]";
	}
```

```
}
```

```
package com.sort;

import java.util.ArrayList;
import java.util.Collections;

public class Test {

     public static void main(String[] args) {

          ArrayList<Employee> al = new ArrayList<Employee>();
          al.add(new Employee(101, "ram", 9000));
          al.add(new Employee(102, "ashok", 3000));
          al.add(new Employee(103, "ajay", 8000));

          Collections.sort(al);

          for(Employee emp : al) {
               System.out.println("id>>" + emp.getId() + "
name>>" + emp.getName() + " salary" + emp.getSalary());
          }

     }
}
```

Output-

```
id>>102 name>>ashok salary>>3000
id>>103 name>>ajay salary>>8000
id>>101 name>>ram salary>>9000
```

Comparator interface-

1. We can use comparator to define our own sorting (customized sorting order)
2. It present in java.util package.
3. It defines two methods-
   Public int compare(Object obj1, Object obj2)
   Public Boolean equals();

4. Whenever we are implementing the comparator interface, compulsory we should provide the implementation for compare() method.
5. Implementing the equals method is optional, because it is already available in every java class from object class.

Example-

```
package com.comparator;

public class Student {

	int id;
	String name;
	int salary; // custom objects

	public Student(int id, String name, int salary) {
		super();
		this.id = id;
		this.name = name;
		this.salary = salary;
	}

	public int getId() {
		return id;
	}

	public void setId(int id) {
		this.id = id;
	}

	public String getName() {
		return name;
	}
```

```
	public void setName(String name) {
		this.name = name;
	}

	public int getSalary() {
		return salary;
	}

	public void setSalary(int salary) {
		this.salary = salary;
	}

	@Override
	public String toString() {
		return "Employee [id=" + id + ", name=" + name + ",
salary=" + salary + "]";
	}

}
```

```
package com.comparator;
import java.util.ArrayList;
import java.util.Collections;


public class Test {


	public static void main(String[] args) {


		ArrayList<Employee> al = new ArrayList<Employee>();
		al.add(new Employee(101, "ram", 9000));
		al.add(new Employee(102, "ashok", 3000));
		al.add(new Employee(103, "ajay", 8000));
```

```
		Collections.sort(al, new NameComparator());


		for (Employee emp : al) {
			System.out.println("id>>" + emp.getId() + " name>>" +
emp.getName() + " salary>>" + emp.getSalary());
		}


	}
}
```

Sort data by using name

```
package com.comparator;


import java.util.Comparator;

public class NameComparator implements Comparator<Employee>
{

	@Override
	public int compare(Employee emp1, Employee emp2) {
		return emp1.name.compareTo(emp2.name);
	}

}
```

Sort data by using salary

```
package com.comparator;

import java.util.Comparator;

public class SalaryComparator implements Comparator<Student>
{

	@Override
```

```
    public int compare(Student employee1, Student employee2)
{

        if (employee1.salary == employee2.salary)
            return 0;
        else if (employee1.salary > employee2.salary)
            return 1;
        else
            return -1;
    }

}
```

Comparable	Comparator
It is meant for default natural sorting order	It is meant for customized sorting order
It is present in java.lang package	It is present in java.util package
This interface defines only one method compareTo()	This interface defines two methods i.e compare() and equals()