Comparable Interface

UNIVERSITY OF CALGARY

CPSC 319 - Data Structures

Benyamin Bashari

Interface

- Abstract Type
- Includes only the signature of methods
- It also can contain final variables
- Every class that implement a Interface, guarantees that they would provide body for the methods of the interface.

Example

```
interface Joinable<T> {
    public T join (T other);
}
```

- Ever class implement this interface can join their object together

Example

```
class Circle implements Joinable<Circle> {
   private int radius;
   public Circle(int radius) {
       this.radius = radius;
   }

   public int getRadius() {
       return radius;
   }

   @Override
   public Circle join(Circle other) {
       return new Circle(radius+other.getRadius());
   }

}
```

Example

```
public class InterfaceExample {

  public static void main(String[] args) {
      Circle a = new Circle(10);
      Circle b = new Circle (20);
      Circle c = a.join(b);
      System.out.println(c.getRadius());
  }
}
```

- **Now we also can write methods that works only with joinable objects.**

Example

```
public static Object joinAll(ArrayList<Joinable> v) {
  if(v.size() == 0)
     return null;
  Object ret = v.get(0);
  for(int i = 1 ; i < v.size() ; i++) {
     Joinable tmp = (Joinable) ret;
     ret = tmp.join(v.get(i));
  }
  return ret;
}

public static void main(String[] args) {
  Joinable a = new Circle(10);
  Joinable b = new Circle (20);
  Joinable c = new Circle (55);
  ArrayList<Joinable> arr = new ArrayList<>();
  arr.add(a);
  arr.add(b);
  arr.add(c);
  Circle ret = (Circle) joinAll(arr);
  System.out.println(ret.getRadius());
}
```

Comparable Interface

```
public interface Comparable<T>
{
  /**
   * Compares this object with another, and returns a numerical result based
   * on the comparison.  If the result is negative, this object sorts less
   * than the other; if 0, the two are equal, and if positive, this object
    * sorts greater than the other


          int compareTo(T o);
 }
```

Comparable Interface

```
class Circle implements Joinable<Circle>, Comparable<Circle>{
  private int radius;
  public Circle(int radius) {
     this.radius = radius;
  }

  public int getRadius() {
     return radius;
  }

  @Override
  public Circle join(Circle other) {
     return new Circle(radius+other.getRadius());
  }

  @Override
  public int compareTo(Circle circle) {
     return radius - circle.radius;
  }
}
```

Comparable Interface

```
public static void main(String[] args) {

   Comparable a = new Circle(7);
   Comparable b = new Circle(15);
   System.out.println(a.compareTo(b));
}
```

Sorting

```
public static void main(String[] args) {

        Circle a = new Circle(22);
        Circle b = new Circle (1);
        Circle c = new Circle (17);
        ArrayList<Circle> arr = new ArrayList<>();
        arr.add(a);
        arr.add(b);
        arr.add(c);
        Collections.sort(arr);
        for(int i = 0 ; i < arr.size() ; i++)
          System.out.println(arr.get(i).getRadius());

}
```

Exercise

- Read first name and last name of each person from command line (Multi Line available in input.txt)
    - N’
    - firstName1 lastName1
    - firstName2 lastName2 …
    - firstNameN lastNameN’
- Make a class Person which implements Comparable
- Override Comparable to compare persons based on first name then last name
- Use sorting in java to sort persons based on their first name