```
// Goal:
int sum = 0;
for(Integer i: new Range(0, 100)) {
    sum += i;
}
```

```
// Idea:
Integer sum(Iterable<Integer> iterable) {
    int sum = 0;
    for(Integer i         : iterable)          {
        sum += i;
    }
    return sum;
}
```

```
interface Function<          T,R >

Type Parameters:
T  -   the type of the input to the function
R  -   the type of the result of the function

Represents a function that accepts one argument and produces a result.

This is a functional interface whose functional method is
apply(Object).

R apply(T t)
Applies this function to the given argument.

class     AddThree    implements Function<Integer, Integer> {
    public Integer apply(Integer value) {
        return value + 3;
    }
}

Function< Integer, Integer > adder = new                    AddThree  ();
Integer result =            adder.apply     ((Integer) 4);
```

```
class Transformed<E> implements             Iterable     <E> {
    class     TransformedIterator           implements Iterator<E> {



        public E next() {


        }

        public     boolean     hasNext  () {


        }
    }

    Function<E, E> transformer;
    Iterable     <E>  iter    ;

    public Transformed(         Iterable     <E>  iter    , Function<E, E> f) {
        this.iter          =  iter    ;
        this.transformer           = f;
    }

    public Iterator<E> iterator() {


    }
}
```

```
class Range implements Iterable<Integer> {
  int currentIndex, low, high;
  /* ... */
}
```

```
class Range implements Iterator<Integer> {
  int currentIndex, low, high;
  /* ... */
}
```

```
class Range implements Iterable<Integer> {
  int low, high;
  /* ... */
}
```

```
class Range implements Iterator<Integer> {
  int low, high;
  /* ... */
}
```