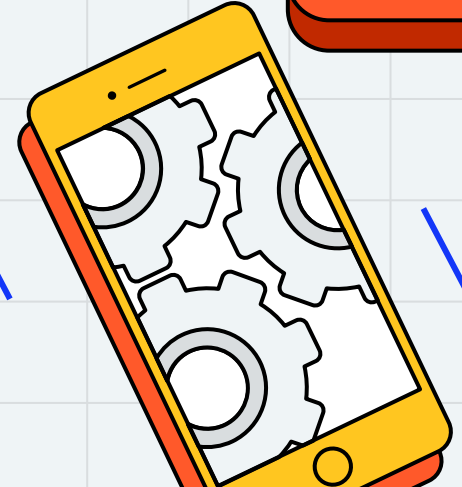


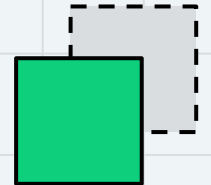


Application

Programming



Hend Alkittawi





Java Iteration

Iterating through objects in Java
Collections

INTRODUCTION

- Related to the discussion on Java generics and collections is the discussion on the following interfaces
 - Iterator, Iterable, Comparable, Comparator
- The Java API has a consistent approach to iterators that are implemented by nearly all collections in the class Library.
- Iterators are implemented in the Java API using two primary interfaces:
 - **Iterator**: used to define an object that can be used as an iterator.
 - **Iterable**: used to define a collection from which an iterator can be extracted.
- The **Comparable** and **Comparator** interfaces in Java facilitate comparisons between objects

THE ITERATOR INTERFACE

- The `Iterator` interface is defined in the Java APIs.
- The interface is used by a class that represents a collection of objects, providing a means to move through the collection one object at a time.
- An **Iterator** provides a consistent and simple mechanism for systematically processing a group of objects.
- An **Iterator** is an object that has methods that allow you to process a collection of items one at a time.
- An **Iterator object** in Java is defined using the **Iterator interface**.

THE ITERATOR INTERFACE

- Every iterator object has a method called **hasNext()** that returns a boolean value indicating whether there is at least one more item to process.
- Every Iterator also has a method called **next()** to retrieve the next item in the collection to process.
- The **Iterator** interface also has a method called **remove()** which takes no parameters and has a void return type. A call to the **remove()** method removes the object that was most recently returned by the **next** method from the underlying collection.

THE ITERABLE INTERFACE

- The Iterable interface has a single method `iterator()` that **returns an Iterator object**.
- If an object has implemented the Iterable interface, we can use a variation of the for loop to process items using a simplified syntax → The enhanced for loop (for-each loop).
- For example if `bookList` is an Iterable object that contains book objects we can use a for loop to process each book object as follows

```
for (Book myBook: bookList)
    system.out.println( myBook)
```

- This version of the for Loop processes each object in the Iterator in turn. It is equivalent to the following:

```
Book myBook;
while( bookList.hasNext()){
    myBook = bookList.next();
    System.out.println( myBook) }
```

```
import java.util.Iterator;

public class Range implements
    Iterable<Integer>{

    private int start, end;

    public Range(int start, int end) {
        this.start = start;
        this.end = end;
    }

    public Iterator<Integer> iterator(){
        return new RangeIterator(start,
            end);
    }
}
```

```
import java.util.Iterator;
import java.util.NoSuchElementException;

public class RangeIterator implements
    Iterator<Integer>{

    private int cursor;
    private int end;

    public RangeIterator(int start, int
        end){

        this.cursor = start;
        this.end = end;
    }

    public boolean hasNext() {
        return this.cursor < end;
    }

    public Integer next() {
        if(this.hasNext()) {
            int current = cursor;
            cursor++;
            return current;
        }
        throw new NoSuchElementException();
    }

    public void remove() {
        throw new
            UnsupportedOperationException();
    }
}
```

```
import java.util.Iterator;

public class RangeTest {

    public static void main(String[] args){
        Range range = new Range(1, 7);
        System.out.println("Looping with an
            iterator");

        Iterator<Integer> it =
            range.iterator();

        while(it.hasNext()){
            int cur = it.next();
            System.out.print(cur + "\t");
        }

        System.out.println("\nLooping with a
            for-each loop");

        for(Integer cur : range){
            System.out.print(cur + "\t");
        }
    }
}
```

```
Looping with an iterator
1   2   3   4   5   6
Looping with a for-each loop
1   2   3   4   5   6
```

```

public class Course {
    private String prefix;
    private int number;
    private String title;
    private String grade;

    public Course(String prefix, int number,
                  String title, String grade){
        this.prefix = prefix;
        this.number = number;
        this.title = title;
        if (grade == null)
            this.grade = "";
        else
            this.grade = grade;
    }

    public Course(String prefix, int number,
                  String title){
        this(prefix, number, title, "");
    }

    public boolean taken(){
        return !grade.equals("");
    }

    public String toString(){
        String result = prefix + " " + number
            + ": " + title;

        if (!grade.equals(""))
            result += " [" + grade + "];"
        return result;
    }
}

```

```

public class ProgramOfStudy implements
    Iterable<Course>{
    private List<Course> list;

    public ProgramOfStudy(){
        list = new LinkedList<Course>();
    }

    public void addCourse(Course course){
        if (course != null)
            list.add(course);
    }

    public String toString(){
        String result = "";
        for (Course course : list)
            result += course + "\n";
        return result;
    }

    @Override
    public Iterator<Course> iterator() {
        return list.iterator();
    }

    public void loadCourses() {
        list.add(new Course("CS", 3443,
            "Application Programming", "A+"));
        list.add(new Course("CS", 3343,
            "Algorithms", "B"));
        list.add(new Course("CS", 1173, "Data
            Analysis and Visualization", "C+"));
        list.add(new Course("CS", 2073,
            "Introduction to Programming"));
    }
}

```

```

public class IterableTest {

    public static void main(String[] args) throws
        Exception{
        ProgramOfStudy pos = new ProgramOfStudy();
        pos.loadCourses();
        System.out.println(pos);

        for(Course course : pos) {
            pos.addCourse(new Course("MATH", 1044,
                "Calculus I"));
        }

        System.out.println("Removing courses with
            no grades.\n");

        Iterator<Course> itr = pos.iterator();

        while (itr.hasNext()){
            Course course = itr.next();
            if (!course.taken())
                itr.remove();
        }
        System.out.println(pos);
    }
}

```




THANK

YOU!

DO YOU HAVE ANY QUESTIONS?



hend.alkittawi@utsa.edu



By Appointment



Online