

Objectives

- Packages
- Collections
- Traversing Collections

1

Reflection: Assignment 5

- Bringing together a variety of concepts:
 - Inheritance
 - Abstract classes
 - Dynamic dispatch/polymorphism
 - Final methods
- Leveraging all you have access to, e.g.,
 - What you inherited, parameters, AND their APIs
- My hope: your answers to the design decisions will be easy for you to express because you understand them well

2

Review

1. How do we specify that a class/method cannot be subclassed/overridden, respectively?
2. What is the keyword for specifying that your class adheres to an interface?
3. What are the 3 components of the Java Collection Framework?
4. What data types can collections hold?
5. How can we convert a primitive type into its respective wrapper class type?
6. What is the syntax to say what type the collection holds?
7. Why is the preference to write code as

```
Interface variable = new Implementation();  
Example: List<Card> hand = new ArrayList<>();
```

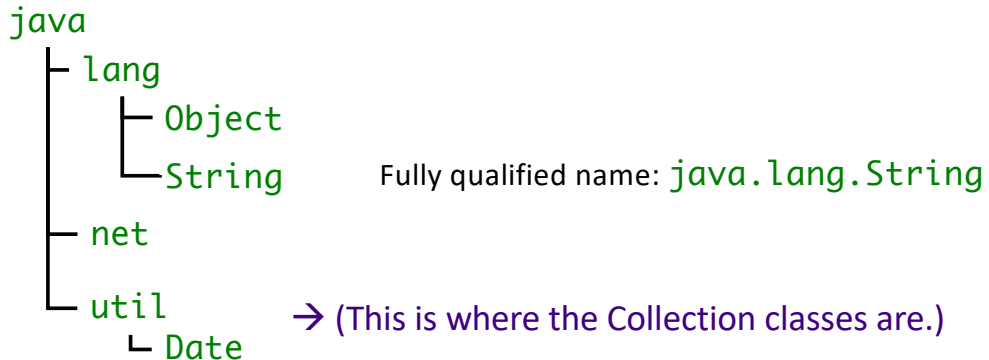
8. What Collection interface, implementations did we discuss?

PACKAGES

Review: Packages

- Hierarchical structure of Java classes

- Directories of directories



- Use `import` to access packages

Oct 6, 2021

Sprenkle - CSCI209

5

5

Importing Packages

- Can import one class at a time or all the classes within a package
- Examples:

```
import java.util.Date;
import java.io.*; ← Import entire package
```

- * form may increase compile time
 - BUT, no effect on run-time performance

Oct 6, 2021

Sprenkle - CSCI209

6

6

Packaging Code

- To reduce chance of a conflict between names of classes, put classes in **packages**
- Use **package** keyword to say that a class belongs to a package:
 - `package java.util;`
 - *First line in class file*
- Typically, use a unique prefix, similar to domain names
 - `com.ibm`
 - `edu.wlu.cs.logic`
- Organize code by the packages
 - For example, code in `edu.wlu.cs.logic` package would be in a `logic` directory inside a `CS` directory inside a `WLU` directory inside a `edu` directory

Oct 6, 2021

We will start organizing our code in packages soon

7

7

Review: Collections Framework

- **Interfaces**
 - Abstract data types that represent collections
 - Collections can be manipulated *independently* of implementation
- **Implementations**
 - Concrete implementations of collection interfaces
 - Reusable data structures
- **Algorithms**
 - Methods that perform useful computations on collections, e.g., searching and sorting
 - Reusable functionality
 - **Polymorphic**: same method can be used on many different implementations of collection interface

Oct 6, 2021

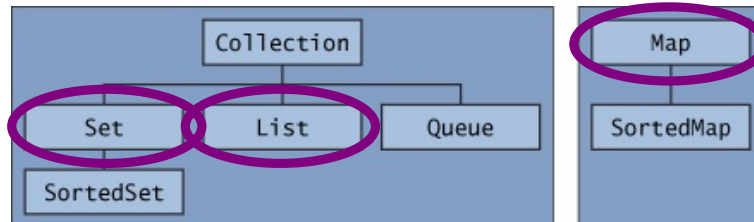
Sprenkle - CSCI209

8

8

Review: Core Collection Interfaces

- Encapsulate different types of collections



```
public abstract class AbstractList<E> extends
AbstractCollection<E> implements List<E>
```

Oct 6, 2021

Sprenkle - CSCI209

9

9

Comparing: Before & After Generics

- Before Generics

```
List myList = new LinkedList();
myList.add(new Card(4, "clubs"));
...
Card x = (Card) myList.get(0);
```

- After Generics

```
List<Card> myList = new LinkedList<>();
myList.add(new Card(4, "clubs"));
...
Card x = myList.get(0);
```

✓ Improved readability and robustness

Oct 6, 2021

Sprenkle - CSCI209

10

10

LISTS

Oct 6, 2021

Sprenkle - CSCI209

11

11

Review: Lists

- Interface: List
- Common implementations: ArrayList, LinkedList

Oct 6, 2021

Sprenkle - CSCI209

12

12

Discussion of Deck Class

`cards.Deck.java`

Oct 6, 2021

Sprenkle - CSCI209

13

13

SETS

Oct 6, 2021

Sprenkle - CSCI209

14

14

Set Interface

- No duplicate elements
 - Needs to determine if two elements are “logically” the same (`equals` method)
- Models mathematical set abstraction

Oct 6, 2021

Sprenkle - CSCI209

15

15

Declaring Sets

- Like Lists, declare type that Set contains:

```
Set<String> mySet = new HashSet<>();
```

Oct 6, 2021

Sprenkle - CSCI209

16

16

Set Interface

- **boolean** `add(<E> o)`
 - Add to set, only if not already present; returns true if added
- **int** `size()`
 - Returns the number of elements in the set
- And more! (`contains`, `remove`, `toArray`, ...)
- Note: no `get` method -- get #3 from the set?


Oct 6, 2021

Sprenkle - CSCI209

17

17

Some Set Implementations

- | | |
|---|---|
| <ul style="list-style-type: none"> ● HashSet  ➤ Implements set using <i>hash table</i> <ul style="list-style-type: none"> ● add, remove, and contains each execute in $O(1)$ time ➤ Used more frequently ➤ Faster than <code>TreeSet</code> ➤ No ordering | <ul style="list-style-type: none"> ● TreeSet ➤ Implements set using a <i>tree</i> <ul style="list-style-type: none"> ● add, remove, and contains each execute in $O(\log n)$ time ➤ Sorts |
|---|---|

Oct 6, 2021

Sprenkle - CSCI209

18

18

FindDuplicates Problem

- From the array of command-line arguments, identify (i.e. print) the duplicates

HashSet()

Set interface:

- `boolean add(<E> o)`
- `int size()`
- `boolean contains(Object o)`

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

Oct 6, 2021

Sprenkle - CSCI209

19

19

FindDuplicates: One solution

```
public static void main(String args[]) {
    Set<String> s = new HashSet<>();
    for (String a : args) {
        if (!s.add(a)) {
            System.out.println( "Duplicate detected: " + a);
        }
    }
    System.out.println(s.size() + " distinct words detected: "
        + s);
}
```

How much does code changes if s is a TreeSet?

Oct 6, 2021

Sprenkle - CSCI209

20

20

MAPS

Oct 6, 2021

Sprenkle - CSCI209

21

21

Maps

- Python called these *dictionaries*
- Maps keys (of type $\langle K \rangle$) to values (of type $\langle V \rangle$)
- No duplicate keys
 - Each key maps to at most one value

Oct 6, 2021

Sprenkle - CSCI209

22

22

Declaring Maps

- Declare types for both keys and values
- `class HashMap<K, V>`

```
Map<String, Integer> map = new HashMap<>();
```

Keys are Strings

Values are Integers

```
Map<String, List<String>> map = new HashMap<>();
```

Keys are Strings

Values are Lists of Strings

Oct 6, 2021

Sprenkle - CSCI209

23

23

Map Interface

- `<V> put(<K> key, <V> value)`
 - Returns old value that key mapped to
- `<V> get(Object key)`
 - Returns value at that key (or null if no mapping)
- `Set<K> keySet()`
 - Returns the set of keys

And more ...

Oct 6, 2021

Sprenkle - CSCI209

24

24

A few Map Implementations

- HashMap
 - Fast
- TreeMap
 - Sorting
 - Key-ordered iteration
- LinkedHashMap
 - Fast
 - Insertion-order iteration

Oct 6, 2021

Sprenkle - CSCI209

25

25

ALGORITHMS

Oct 6, 2021

Sprenkle - CSCI209

26

26

Collections Framework's Algorithms

- *Polymorphic algorithms*
- Reusable functionality
- Implemented in the `Collections` class
 - Similar to `Arrays` class, which operates on arrays
 - Static methods, 1st argument is the Collection

Oct 6, 2021

Sprenkle - CSCI209

27

27

Overview of Available Algorithms

- *Sorting* – optional Comparator
 - *Shuffling*
 - *Searching* – `binarySearch`
 - *Routine data manipulation*: `reverse*`, `copy*`, `fill*`, `swap*`, `addAll`
 - *Composition* – frequency, disjoint
 - *Finding min, max*
- } * Only Lists

Oct 6, 2021

Sprenkle - CSCI209

28

28

TRaversing Collections

Oct 6, 2021

Sprenkle - CSCI209

29

29

Traversing Collections: For-each Loop

- For-each loop:

```
for (Object o : collection)
    System.out.println(o);
```

Or whatever data type is appropriate

- Valid for all Collections

➤ Maps (and its implementations) are not Collections

- But, Map's `keySet()` is a Set and `values()` is a Collection

Oct 6, 2021

Sprenkle - CSCI209

30

30

Traversing Lists: Iterator

Not covered during class

- Always between two elements



```
Iterator<Integer> i = list.iterator();  
while( i.hasNext()) {  
    int value = i.next();  
    ...  
}
```

Oct 6, 2021

Sprenkle - CSCI209

31

31

Benefits of Collections Framework

- ?

Oct 6, 2021

Sprenkle - CSCI209

38

38

Benefits of Collections Framework

- **Provides common, well-known interface**
 - Allows interoperability among unrelated APIs
 - Reduces effort to learn and to use new APIs for different implementations
- **Reduces programming effort:** provides useful, reusable data structures and algorithms
- **Increases program speed and quality:** provides high-performance, high-quality implementations of data structures and algorithms; interchangeable implementations → tuning
- **Reduces effort to design new APIs:** use standard collection interface for your collection
- **Fosters software reuse:** New data structures/algorithms that conform to the standard collection interfaces are reusable

Oct 6, 2021

Sprenkle - CSCI209

39

39

Looking Ahead

- **Exam 1 – Friday**
 - **Online, timed exam: 70 minutes**
 - No class Friday
 - Opens: Friday at 8:30 a.m.; Closes: Sunday at 11:59 p.m.
 - **Open book/notes/slides – but **do not** rely on that**
 - NOT open internet
 - **Prep document online**
 - **3 sections:**
 - Very Short Answer, Short Answer, Coding
- **Questions**

Oct 6, 2021

Sprenkle - CSCI209

40

40