Collections

Arrays and lists are straightforward for storing a collection of objects. In this activity, you'll gain experience with two other kinds of collections. Sets and maps are quite useful for implementing a wide variety of algorithms.

Manager: Recorder:

Presenter: Reflector:

Content Learning Objectives

After completing this activity, students should be able to:

- Describe Java's Collections Framework.
- Summarize methods in the `Set` interface.
- Summarize methods in the `Map` interface

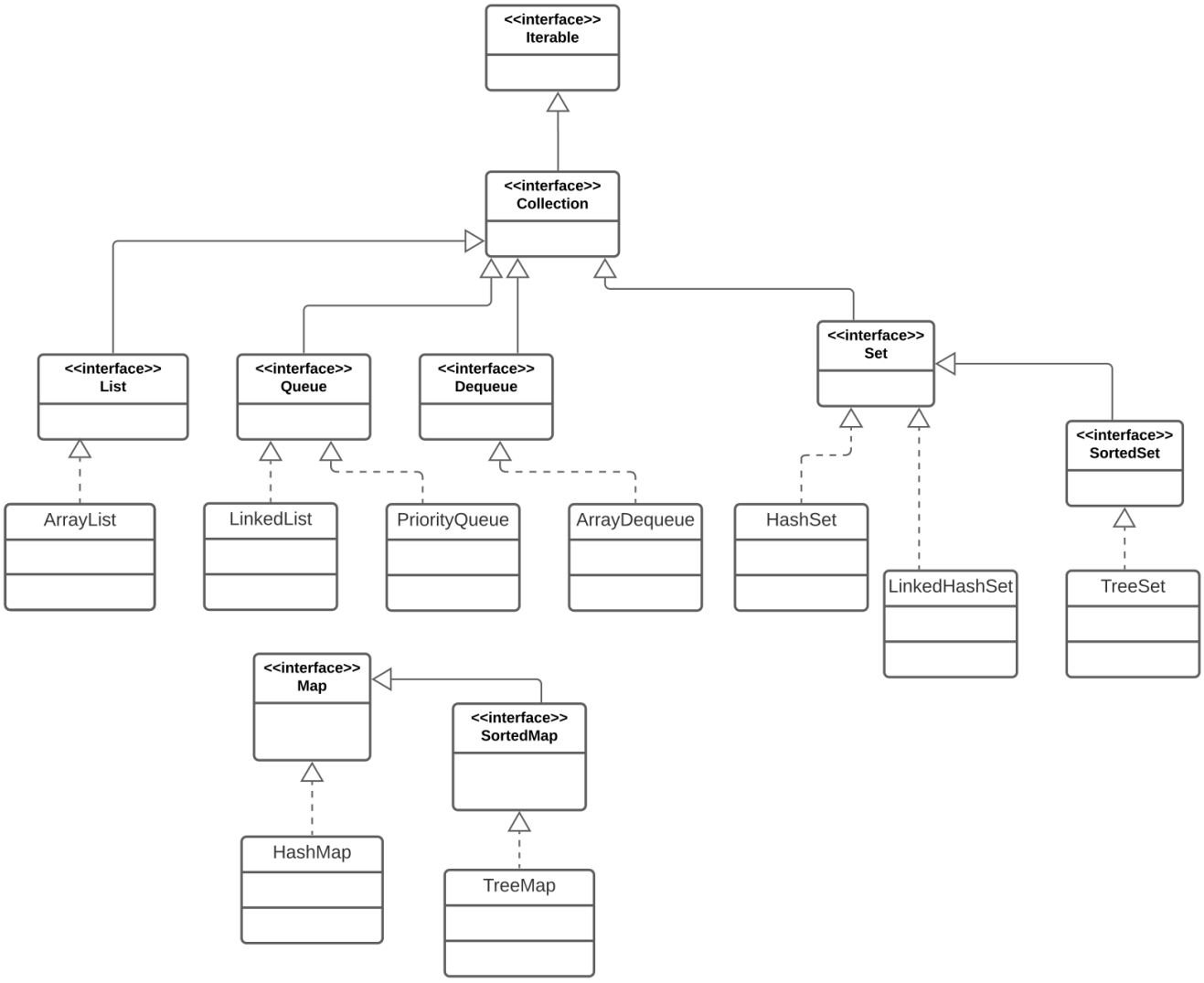
Process Skill Goals

During the activity, students should make progress toward:

- Interpreting results after running code in JShell. (Information Processing)

Model 1 Collections Framework

A collection is an object that represents a group of objects. The Java library includes many types of collections. The UML diagram below shows some of these collections and their relationships.



Questions (10 min) Start time:

1. Based on the diagram, write "extends" or "implements" for each relationship:

a) ArrayList List

b) List Collection

c) HashMap Map

d) SortedMap Map

e) Set Collection

f) TreeSet SortedSet

2. What is the difference between the solid and dashed lines in the diagram?

3. List all the concrete classes in the UML diagram.

4. Based on your prior experience with ArrayList and LinkedList, what are two methods that you expect to be in the Collection interface?

5. Can a HashMap object be assigned to a Collection variable? Justify your answer.

Note: The Collection interface declares the following methods:

`add()`, `addAll()`, `clear()`, `contains()`, `containsAll()`, `equals()`, `hashCode()`, `isEmpty()`, `iterator()`, `parallelStream()`, `remove()`, `removeAll()`, `removeIf()`, `retainAll()`, `size()`, `spliterator()`, `stream()`, and `toArray()`.

Model 2 Set of Strings

Each line of code below depicts the result of the expression in the first column being executed in *JShell*.

Java code	Shell output
`Set<String> names = new Set<>();`	java.util.Set is abstract; cannot be instantiated
`Set<String> names = new HashSet<>();`	[]
`names.add("WAS")`	true
`names.add("BAL")`	true
`names.add("PHI")`	true
`names`	[PHI, WAS, BAL]
`names.contains("DEN")`	false
`names.add("DEN")`	true
`names.contains("DEN")`	true
`names.contains("den")`	false
`names.add("DEN")`	false
`names.add(123)`	int cannot be converted to java.lang.String
`names.size()`	4
`names`	[PHI, WAS, DEN, BAL]
`names.remove("WAS")`	true
`names.remove("IND")`	false
`names`	[PHI, DEN, BAL]
`names.isEmpty()`	false
`names.clear()`	
`names.size()`	0
`names.isEmpty()`	true

Questions (15 min) Start time:

6. For the collection above:

a) What is the interface name?

b) What is the class name?

c) What is the variable name?

d) What is the element type?

7. Based on the shell output, describe what the following methods return:

a) `add`

b) `remove`

8. Consider the contents of `names` just before `"WAS"` was removed.

a) What was the size of `names` at this point?

b) How many times was the `add` method called?

c) Explain why these two numbers are different.

9. Continuing the previous question:

a) In what order were the strings added to the set?

b) In what order were they displayed in the output?

c) Why do you think the two orders are different?

10. In your own words, summarize what a `Set` is in Java. Give an example from everyday life.

11. In discrete mathematics, sets have three basic operations:

- Union ($S \cup T$) : all elements in S or T (or both)
- Intersection ($S \cap T$) : elements in both S and T
- Difference ($S - T$) : elements in S but not in T

Based on the documentation for `java.util.Set`, which methods implement these operations?

Model 3 Map of Team Names

The following abbreviations are for National Football League (NFL) teams:

ATL	Atlanta Falcons
DEN	Denver Broncos
IND	Indianapolis Colts
MIA	Miami Dolphins
SEA	Seattle Seahawks

Java code	Shell output
`Map<String, String> teams;`	null
`teams = new Map<>();`	java.util.Map is abstract; cannot be instantiated
`teams = new HashMap<>();`	{}
`teams.isEmpty()`	true
`teams.put("MIA", "Miami Dolphins")`	null
`teams.put("MIA", "Miami")`	"Miami Dolphins"
`teams.size()`	1
`teams`	{MIA=Miami}
`teams.put("ATL", "Atlanta")`	null
`teams.put("SEA", "Seattle")`	null
`teams`	{MIA=Miami, ATL=Atlanta, SEA=Seattle}
`teams.containsKey("ATL")`	true
`teams.containsKey("DEN")`	false
`teams.containsValue("Miami")`	true
`teams.containsValue("Dolphins")`	false
`teams.get("SEA")`	"Seattle"
`teams.get("IND")`	null
`teams.get(0)`	null
`teams.remove("MIA")`	"Miami"
`teams.remove("MIA")`	null
`teams`	{ATL=Atlanta, SEA=Seattle}
`teams.keySet()`	[ATL, SEA]
`teams.values()`	[Atlanta, Seattle]

Questions (20 min) Start time:

12. For the collection above:

a) What is the interface?

b) What is the class?

c) What type of keys?

d) What type of values?

13. Based on the shell output, describe what the following methods return:

a) `put`

b) `get`

14. What type of object does the `keySet` method return? Describe its contents.

15. What type of object does the `values` method return? Describe its contents.

16. In your own words, summarize what a `Map` is in Java. Give an example from everyday life.

17. Why did `teams.get(0)` return null, even though there were values in the map?

18. Write Java code that defines a map named `dow` that represents the seven days of the week as follows: Sun=1, Mon=2, Tue=3, etc.

19. Print the `dow` variable in *JShell*. What do you notice about the order of its contents?