Topic 06: == operator, equals(), hashCode(), Collection use case and memory concern

== operator vs equals() method

- == is a operator
- equals() is a method
- Default equals() is inherited from Object class
- Default equals() method calls ==
- Programmer must override equals() using the desired variable
- equals() method must be reflexive, symmetric, transitive, consistent, false on null

Read more: https://dzone.com/articles/java-hashcode-and-equals-deep-dive

hashCode()

- Returns a hash code value for the object. This method is supported for the benefit of hash tables such as those provided by HashMap.

- The general contract of hashCode is:
- Whenever it is invoked on the same object more than once during an execution of a Java application, the hashCode method must consistently return the same integer, provided no information used in equals comparisons on the object is modified. This integer need not remain consistent from one execution of an application to another execution of the same application.
- If two objects are equal according to the equals(Object) method, then calling the hashCode method on each of the two objects must produce the same integer result.
- It is not required that if two objects are unequal according to the equals(java.lang.Object) method, then calling the hashCode method on each of the two objects must produce distinct integer results. However, the programmer should be aware that producing distinct integer results for unequal objects may improve the performance of hash tables.

Source: https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html

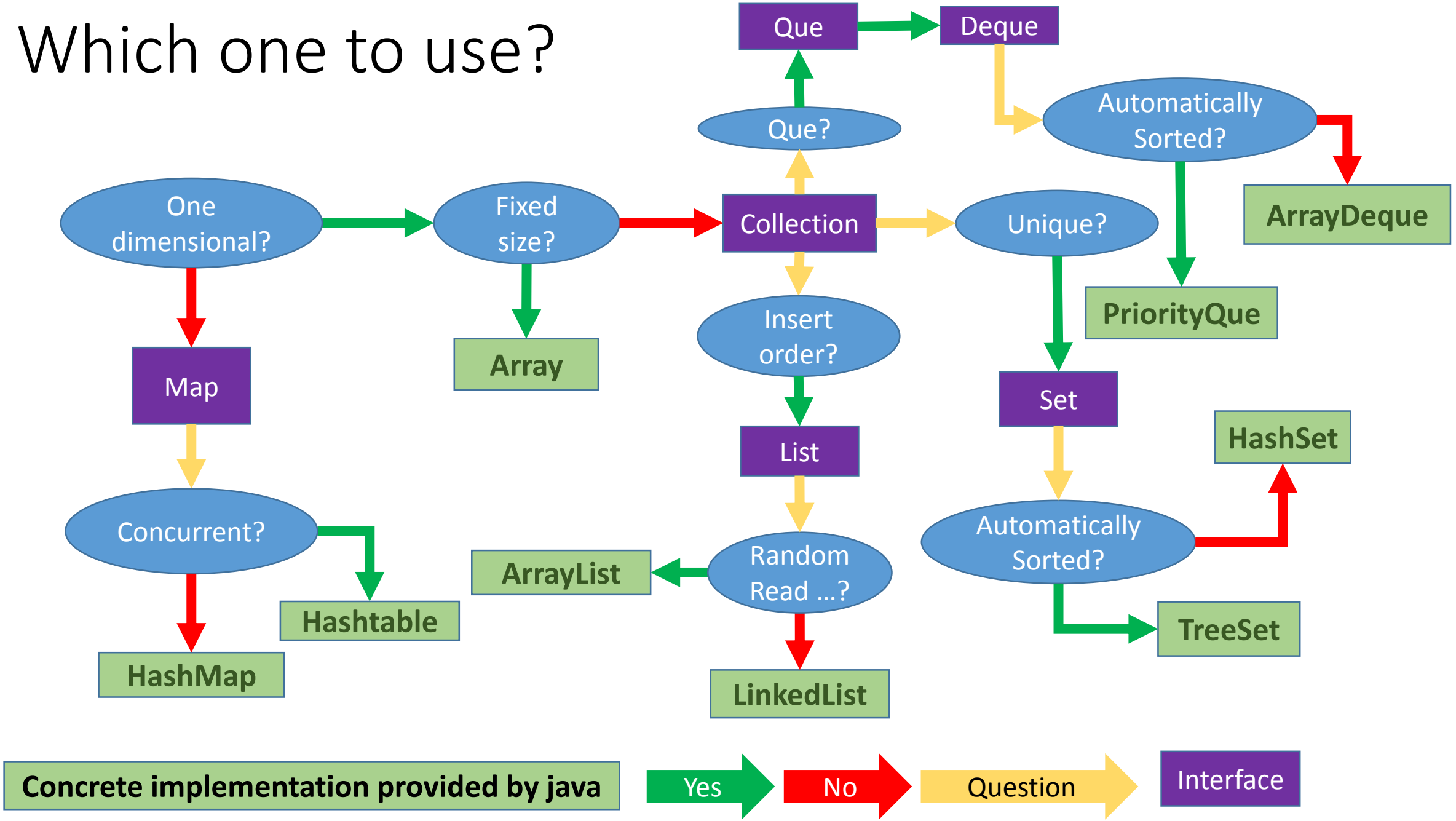
Implementation Task

- Create a class and override equals() and hashCode()
- Test that if it works
- Create a HashSet of few objects where some of them should return true upon calling equals() and make sure HashSet has only unique objects

Collection discussion 2

	ArrayList	HashMap	Hashtable	HashSet	Vector	ArrayDeque	LinkedList
Default size	10	16	11	16	10	8	1
Default Load factor		0.75	0.75	0.75			
Maximum capacity		2^30	ffffff - 8				
Thread safe	No	No	Yes	No	Yes		No

Which one to use?



Time complexity for ArrayList vs LinkedList

	Add	**Remove**	**Get**	**Contains**	**Data Structure**
ArrayList	O(1)	O(n)	O(1)	O(n)	Array
LinkedList	O(1)	O(1)	O(n)	O(n)	Linked List
CopyonWriteArrayList	O(n)	O(n)	O(1)	O(n)	Array

Primitives and wrapper class memory consumption

type	Bytes	type	Byte (32 bit jvm)
boolean	1	Boolean	8 + 4 +1 = ?
byte	1	Byte	8 + 4 +1
char	2	Char	8 + 4 +2
short	2	Short	8 + 4 +2
int	4	Integer	8 + 4 +4
float	4	Float	8 + 4 +4
long	8	Long	8 + 4 +8
double	8	Double	8 + 4 +8

What is default values for the primitive variable?

Over

Concrete Class	Default Capacity	Overhead 10K	Resize	Empty
HashSet	16	360 kB	2x	
HashTable	11	360 kB	2x + 1	
HashMap	16	360 kB	2x	
LinkedList	1	240 kB	+1	48
ArrayList	10	40 kB	1.5x	88
Array	Size declared			