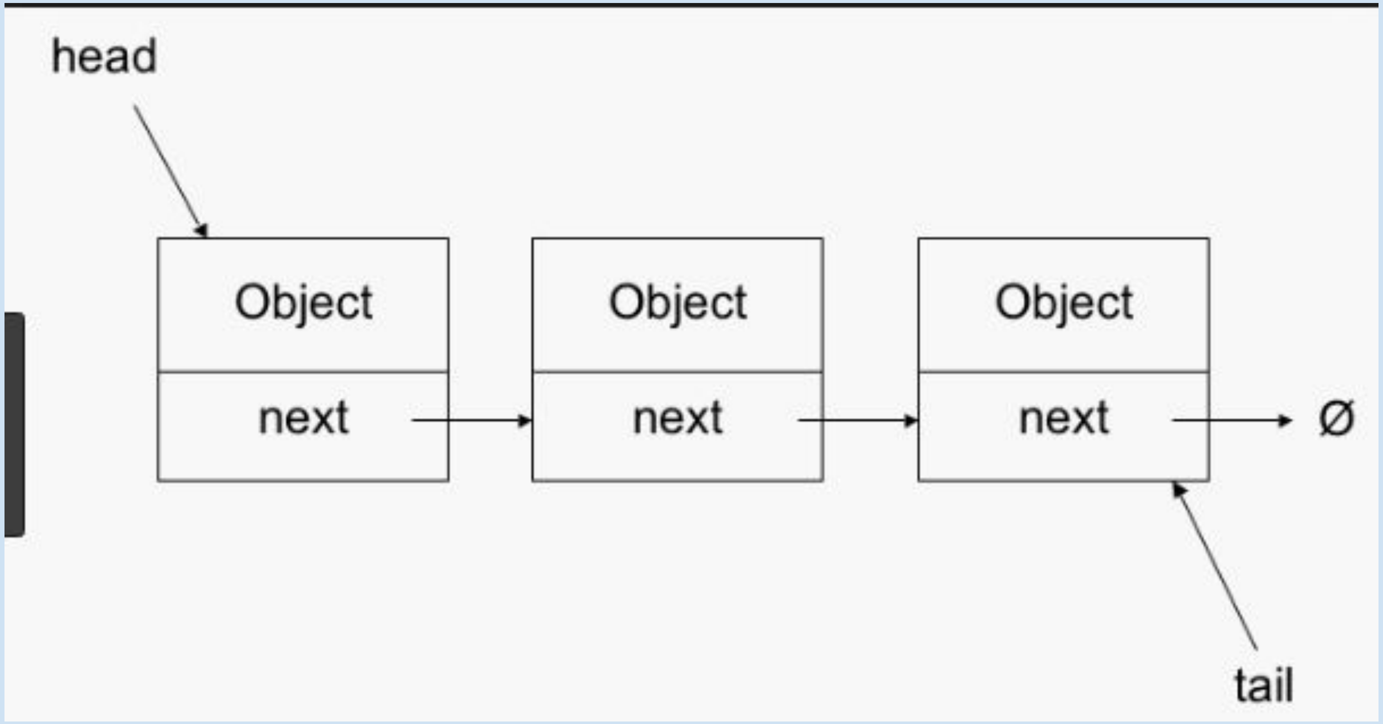
CS 342 Software Design

Today:

- Finish up inner classes
- Final keyword
- Collections
- Finish Generic Programing

Generic Programing: An Example



Java Generic Data Structures: Collections

ArrayList, LinkedList, HashMap, HashSet….

Let’s look at ArrayList:

https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html

What is the output of this code?

```
ArrayList<Integer> myList = new
ArrayList<Integer>();
        myList.add(200);
        myList.add(300);
        myList.add(400);
        myList.add(500);
        myList.remove(2);

        System.out.println(myList.get(3));
```

A) 500
B) 400
C) 300
D) None of the above

Main Benefits of Generic Programing:

Type safe collections: errors caught at compile time.

Code reuse: can write method/interface/class once for any type we want.

Lets try to code it!

How to go through an ArrayList<>:

- Traditional for loop: for(int i = 0; i < myList.size(); i++){}
- Get an Iterator: Iterator<Integer> i = myList.iterator();
- Old for-each loop: for(int val : myList){}
- New For-each: forEach(e->do something);

Which one to use depends on what you want to do*

****What kind of performance do you need***

Clicker Question: What is the output?

```
ArrayList<int> myList = new ArrayList<int>();

myList.add(20);

myList.add(30);

myList.add(40);

myList.remove(1);

System.out.println(myList.get(1));
```

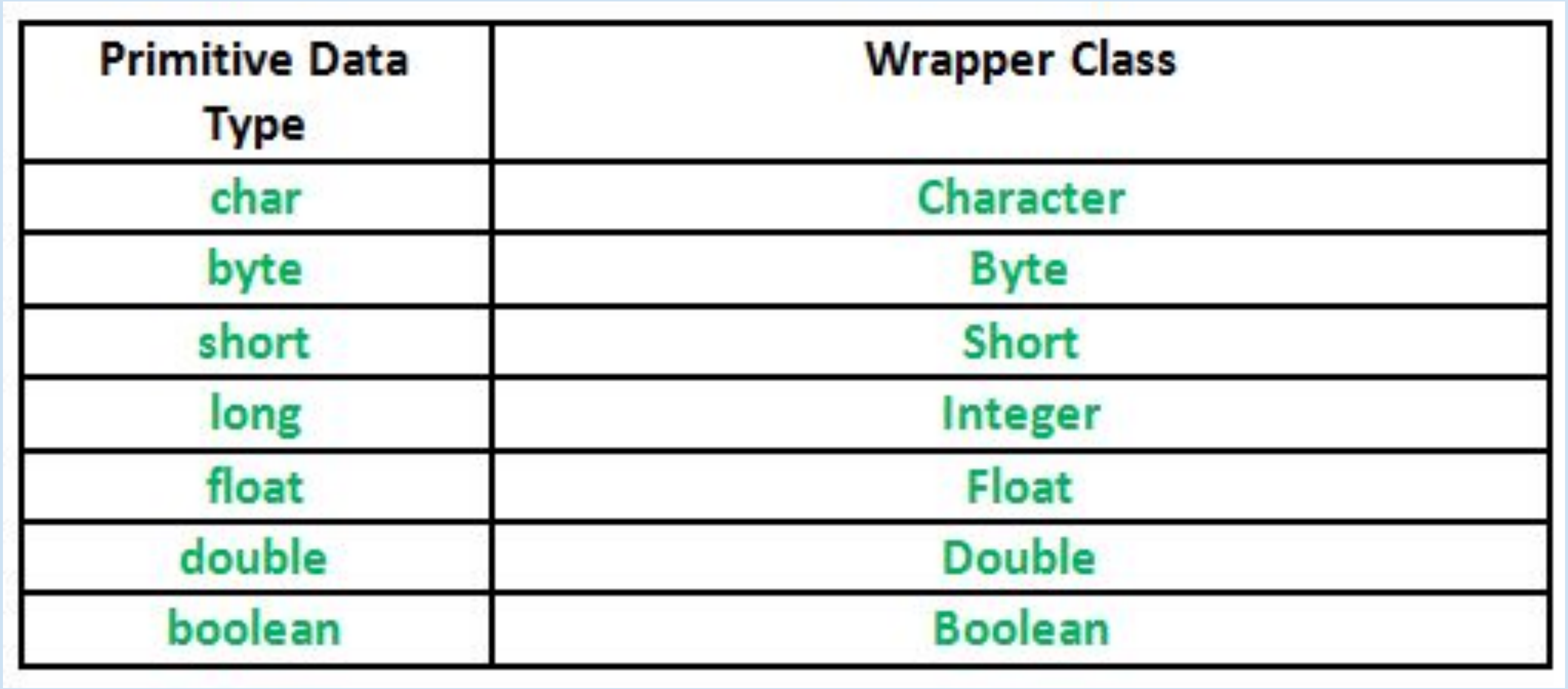
A) 20
B) 30
C) 40
D) Null pointer exception
E) Doesn’t compile

Why Integer instead of int?

- Objects are needed if we wish to modify the arguments passed into a method (because primitive types are passed by value).
- Data structures in the Collection framework, such as ArrayList and Vector, store only objects.
- An object is needed to support synchronization in multithreading.

Wrapper Classes

A class whose object wraps or contains a primitive type



Primitive Data Type	Wrapper Class
char	Character
byte	Byte
short	Short
long	Integer
float	Float
double	Double
boolean	Boolean

Lets Look At Integer!

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

Now lets see an example!

When you see “Wrapper”, think: Java class

When you see “Rapper”, think:

Autoboxing and Unboxing:

Autoboxing: The automatic conversion that the Java compiler makes between the primitive types and their corresponding object wrapper classes

Unboxing: Converting an object of a wrapper type (Integer) to its corresponding primitive (int) value

Clicker Question: Variable Types in Java are divided into two categories; what are they?

A) Primary and Required
B) Referred and Prime
C) Primitive and Reference
D) Reference and Primary
E) Primitive and Required