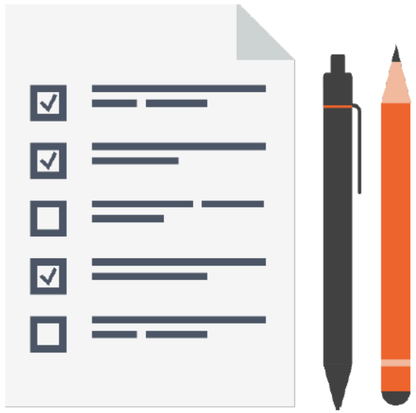
ArrayList and Wrapper Classes



- ArrayList Class
- ArrayList vs Array
- ArrayList methods
- Wrapper Classes
- Autoboxing
- Unboxing

After today's session you should be able to:

- create and use ArrayList
- Utilize arrayList methods
- convert primitives to objects

Wrapper Classes

Java provides wrapper class for each of the primitive types

byte	Byte
boolean	Boolean
char	Character
double	Double
float	Float
int	Integer
long	Long
short	Short

Wrapper Classes

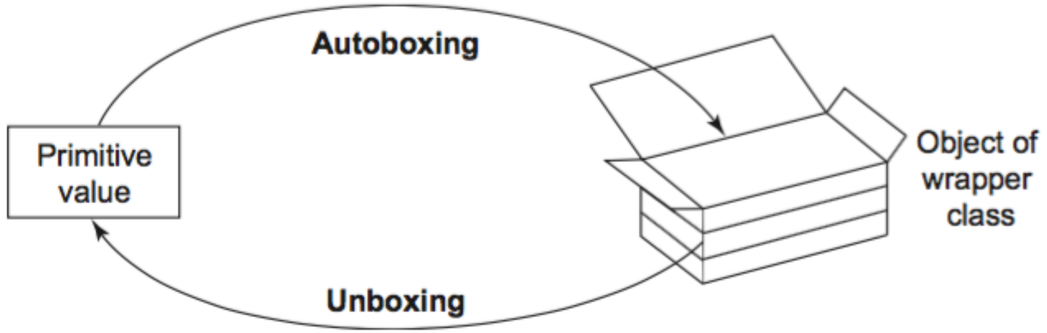
```
Integer name = new Integer();

Character name = new Character();

Boolean name = new Boolean(true);

Byte name = new Byte(11);
```

Autoboxing and Unboxing



CONVERTING STRING > OTHER TYPES

Wrapper class	Converting `String` to primitive	Converting `String` to wrapper class
Boolean	`Boolean.parseBoolean("true");`	`Boolean.valueOf("TRUE");`
Byte	`Byte.parseByte("1");`	`Byte.valueOf("2");`
Short	`Short.parseShort("1");`	`Short.valueOf("2");`
Integer	`Integer.parseInt("1");`	`Integer.valueOf("2");`
Long	`Long.parseLong("1");`	`Long.valueOf("2");`
Float	`Float.parseFloat("1");`	`Float.valueOf("2.2");`
Double	`Double.parseDouble("1");`	`Double.valueOf("2.2");`
Character	None	None

ArrayList Class

```
ArrayList<Base_Type> name = new ArrayList<Base_Type>();
```

```
ArrayList<String> name = new ArrayList<String>();
```

ArrayList can store only objects

ArrayList Class

```
ArrayList<Base_Type> name = new ArrayList<Base_Type>(Capacity);
```

```
ArrayList<String> myArray = new ArrayList<String>(10);
```

ArrayList

Can store only objects

Can be resized

Array

Can store primitives and objects

Has defined size and cannot be changed

ArrayList Methods

`add(Type element)`	Adds the element to the end of this list
`get(int index)`	Returns the element at the specified position
`contains(Object element)`	Returns true if element is in this list; otherwise, returns false
`indexOf(Object element)`	Returns the index of the first occurrence of element in this list
`clear()`	Removes all elements from this list
`size()`	Returns the number of elements in this list
`remove(int index)`	Removes the element at the specified index of this list
`isEmpty()`	Returns true if this list is empty; otherwise, returns false