Wrappers & ArrayLists

Introduction

This unit covers Objects that can take the place of primitives and lists that can be resized.

Wrappers

Wrappers are object versions of primitive values.

The following is a list of Wrappers and what primitives they correspond to:

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

Creating a Wrapper

The process of creating a Wrapper from a primitive value is called **Wrapping** or **Boxing**.

Format for Creating a Wrapper:

WrapperType name = new WrapperType(primitiveValue);

Example:

```
Integer i = new Integer(5);
```

Accessing the primitive value from a Wrapper

The process of getting the primitive value that a Wrapper stores is called **un-wrapping** or **un-boxing**.

Format for un-wrapping a primitive value:

primitiveType name = wrapperName.unwrappingMethod();

Example:

(assume i is an Integer)

```
int x = i.intValue();
```

Auto-Wrapping and Auto-Un-Wrapping

In most cases java will automatically wrap up and un-wrap values for you. This occurs when you try to use a wrapper as a primitive or a primitive as a wrapper.

Example of auto wrapping

```
Integer x = 55;
```

Example of auto un-wrapping

```
Integer y = new Integer(17);
int z = y;
```

Important Wrapper Methods

Character

Method	Description	Example

charValue()	Returns the char value the wrapper is storing.	Chacter c = new Character(‘a’);
		char letter = c.charValue();

boolean isDigit(char c)	Returns true if the character is a number	char a = ‘6’; char b = ‘a’; System.out.println(Character.isDigit(a)); System.out.println(Character.isDigit(b)); **Output:** true false
boolean isLetter(char c)	Returns true if the character is a letter	char a = ‘6’; char b = ‘a’; System.out.println(Character.isLetter(a)); System.out.println(Character.isLetter(b)); **Output:** false true
boolean isLetterOrDigit(char c)	Returns true if the character is a digit or letter	char a = ‘ ’; char b = ‘a’; System.out.println(Character.isLetterOrDigit(a)); System.out.println(Character. isLetterOrDigit (b)); **Output:** false true
boolean isLowerCase(char c)	Returns true if the character is a lowercase letter	char a = ‘A’; char b = ‘a’; System.out.println(Character. isLowerCase (a)); System.out.println(Character. isLowerCase (b)); **Output:** false true
boolean isUpperCase(char c)	Returns true if the character is an uppercase letter	char a = ‘A’; char b = ‘a’; System.out.println(Character. isUpperCase (a)); System.out.println(Character. isUpperCase (b)); **Output:** true false

boolean isSpace(char c)	Returns true if the character is a space	char a = ‘ ’; char b = ‘a’; System.out.println(Character. isSpace (a)); System.out.println(Character. isSpace (b)); **Output:** true false
boolean isWhitespace(char c)	Returns true if the character is a white space(‘\n’,’\t’,’ ‘)	char a = ‘ ’; char b = ‘a’; System.out.println(Character. isWhitespace (a)); System.out.println(Character. isWhitespace (b)); **Output:** true false
char toLowerCase(char c)	Returns the lowercase version of the character	char a = ‘A’; char b = ‘a’; System.out.println(Character. toLowerCase (a)); System.out.println(Character. toLowerCase (b)); **Output:** a a
char toUpperCase(char c)	Returns the uppercase version of the character	char a = ‘A’; char b = ‘a’; System.out.println(Character. toUpperCase (a)); System.out.println(Character. toUpperCase (b)); **Output:** A A

Byte

Method	**Description**	**Example**

byte byteValue()	Returns the byte value the wrapper is storing.	Byte a = new Byte(9);
		byte smallNumber = a.byteValue();

byte parseByte(String s)	Returns the byte value stored in a String	String a = “5”; byte b= Byte.parseByte(a); System.out.println(b); **Output:** 5

String toString(byte i,int radix)	Returns a String that contains the value i in the base of radix	byte a = 5;
		String base2 = Byte.toString(a,2);
		System.out.println(base2);
		
		Output:
		101

Short

Method	Description	Example
short shortValue()	Returns the short value the wrapper is storing.	Short s = new Short(159); short t = s.shortValue();
short parseShort(String s)	Returns the short value stored in a String	String a = “445”; short b= Short.parseLong(a); System.out.println(b); **Output:** 445
String toString(short i,int radix)	Returns a String that contains the value i in the base of radix	byte a = 11; String base16 = Short.toString(a,16); System.out.println(base16); **Output:** B

Integer

Method	Description	Example
int intValue()	Returns the int value stored by the wrapper.	Integer i = new Integer(1234); int num = i.intValue();
int parseInt(String s)	Returns the int value stored in a String	String a = “78”; int b= Integer.parseInt(a); System.out.println(b); **Output:** 78
String toString(int i,int radix)	Returns a String that contains the value i in the base of radix	int a = 15; String base8 = Integer.toString(a,8); System.out.println(base2); **Output:** 17

Long

Method	Description	Example
long longValue	Returns the long value stored by the wrapper.	Long i = new Long(1112233444); long num = i.intValue();
long parseLong(String s)	Returns the long value stored in a String	String a = “985345”; long b= Long.parseLong(a); System.out.println(b); **Output:** 985345
String toString(long i,int radix)	Returns a String that contains the value i in the base of radix	long a = 5; String base2 = Long.toString(a,2); System.out.println(base2); **Output:** 101

Double

Method	Description	Example

double doubleValue()	Returns the double value stored by the wrapper.	Double money = new Double(4.35);
		double m = money.doubleValue();

double parseDouble(String s)	Returns the double value stored in a String	String a = “86.5”; double b= Double.parseDouble(a); System.out.println(b); **Output:** 86.5

Float

Method	Description	Example

float floatValue()	Returns the float value stored by the wrapper.	Float f = new Float(55342.67);
		float income = f.floatValue();

float parseFloat(String s)	Returns the float value stored in a String	String a = “86.5”; float b= Float.parsefloat(a); System.out.println(b); **Output:** 86.5

Boolean

Method	Description	Example

boolean booleanValue()	Returns the boolean value stored by the wrapper.	Boolean answer = new Boolean(true);
		boolean b = answer.booleanValue();

boolean parseBoolean(String c)	Returns the boolean value stored by the String	String a = “false”; double b= Boolean.parseBoolean(a); System.out.println(b); **Output:** false

ArrayList

ArrayLists are objects that store multiple values just like arrays, but with two main differences; the size of an ArrayList can change after it has been created and ArrayLists can only hold Objects.

Importing ArrayList:

In order to use the ArrayList class you must add the following import at the top of your program:

```
import java.util.ArrayList;
```

Format for creating an ArrayList Objects:

ArrayList <u>name</u> = new ArrayList();

Format for creating an ArrayList of a set type:

ArrayList<<u>ObjectType</u>> <u>name</u> = new ArrayList<<u>ObjectType</u>>();

Example:

```
ArrayList<Integer> numbers = new ArrayList<Integer>();
```

ArrayList Methods

Note: E represents the type of Data the ArrayList stores

Method	Description
size()	returns an int value of the number of elements the ArrayList stores
toArray()	returns an array form of the ArrayList
get(int index)	returns the element at the given index
remove(int index)	Removes the element at the given location and returns its value.
set(int index, E value)	Replaces the value at the given index with the sent object. The replaced value is returned.
remove(E value)	Removes the 1st occurrence of the value from the ArrayList. Returns true if the item was found and removed, otherwise it returns false.
contains(E values)	Returns true if the item was found or false if it was not.
add(E value)	Adds the given value to the end of the ArrayList.
add(int index, E value)	Adds the given value by placing it at the given index; items are shifted right to make room for this new value.
clear()	Removes all values from the ArrayList.
indexOf(E value)	Returns the index of the given value or -1 if the value was not found.
isEmpty()	Returns true if the size of the ArrayList is 0 and false if it is not.

Example:

```
ArrayList<Integer> numbers = new ArrayList<Integer>();
numbers.add(5);
numbers.add(6);
numbers.add(7);
numbers.add(1,88);
System.out.println(numbers);
System.out.println(numbers.remove(3));
System.out.println(numbers.set(0,15));
System.out.println(numbers.get(2));
System.out.println(numbers);
```

Output:

```
[5, 88, 6, 7]
7
5
6
[15, 88, 6]
```

For Each and ArrayLists

For each loops can be used to navigate through ArrayLists or to change stored values. Adding or removing from an ArrayList while using a for each loop will cause an error.

Removing Items

When making a for loop to remove items from an ArrayList it is good to start from the end of the list and work down to location 0, otherwise it is likely that locations will be skipped.

Example 1:

```
ArrayList<Integer> nums = new ArrayList<Integer>();
nums.add(2);
nums.add(7);
nums.add(9);
nums.add(3);

// code that attempts to remove all values greater than 5
for(int x=0; x<nums.size(); x++)
    if(nums.get(x) > 5)
        nums.remove(x);
System.out.println(nums);
```

Output:

```
[2, 9, 3]
```

Explanation:

When x is 1 7 is removed for being too large and then the index is changed to 2. 9 is at spot 1 after the removal and never gets processed.

Example 2:

```
ArrayList<Integer> nums = new ArrayList<Integer>();
nums.add(2);
nums.add(7);
nums.add(9);
nums.add(3);

// code that attempts to remove all values greater than 5
for(int x= nums.size()-1; x>=0; x--)
        if(nums.get(x) > 5)
                nums.remove(x);
System.out.println(nums);
```

Output:

```
[2, 3]
```

Explanation:

When x is 2, 9 is removed for being too large and then the x is changed to 1. When x is 1, 7 is removed.

Blue Pelican Sections

Lesson 21
Lesson 42
Lesson 43

Terms

ArrayList	A data structure that stores Objects and can be resized.
Auto Boxing	When java automatically unwraps a wrapper.
Auto Un-boxing	When java automatically wrappers a primitive into a wrapper.
Boxing / Wrapping	Storing a primitive into a wrapper
Un-Boxing / Un-wrapping	Getting the primitive value from a wrapper.
Wrappers	Object versions of primitives.