Object Oriented Programming using Java

Prepared By:
Suyel, PhD
Assistant Professor
Dept. of CSE, NIT Patna

Outline

1. Wrapper Classes

2. Methods of Wrapper Classes

3. Autoboxing and Unboxing

Wrapper Classes

- The wrapper class in Java provides the mechanism to convert primitive data type into object and object into primitive data type.
- A wrapper class is a class, whose object wraps or contains a primitive data type.
- Wrapper classes in Java are used to convert primitive types (int, char, float, etc.) into corresponding objects.
- When we create an object to a wrapper class, it contains a field. In this field, we can store a primitive data type value.
- All the wrapper classes in Java are immutable.

Wrapper Classes (Cont…)

- **Need of Wrapper Classes**
  - We need object, when we want to change the arguments passed into a method (call by value). Wrapper classes convert primitive data types into objects.
  - The classes of java.util package only deal with objects. Wrapper classes help us to make object.
  - To support synchronization with objects in Multithreading.
  - Java collection framework (ArrayList, LinkedList, PriorityQueue, etc.) works with objects only.
  - We need to convert the objects into streams to perform the serialization. If we have a primitive value, we can convert it into objects through the wrapper classes.

Wrapper Classes (Cont…)

- The eight classes of the *java.lang* package are known as wrapper classes in Java.

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

Methods of Wrapper Classes

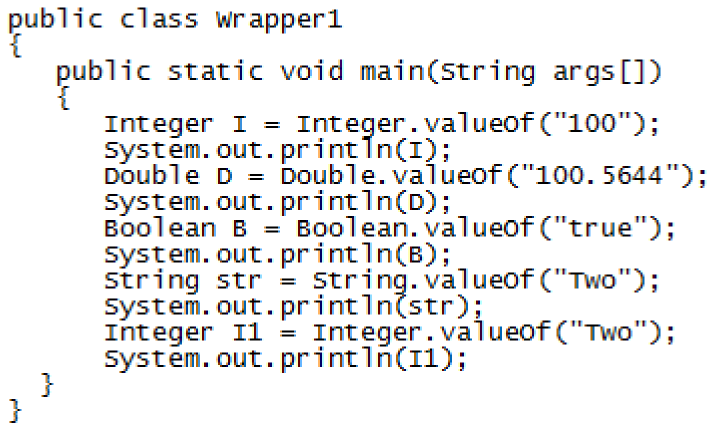
❑ **valueOf() method**

- It is a static method.
- We can also use valueOf() method to convert primitive data types into corresponding objects.
- There are three forms of valueOf().

1. **Wrapper valueOf(String str):** Each wrapper class except Character class contains a static valueOf() method to create Wrapper class object for a given string. **Syntax:** public static Wrapper valueOf(String str);
2. **Wrapper valueOf(String str, int radix):** Every integral Wrapper class (Byte, Short, Integer and Long) contains this valueOf() method to create a Wrapper object for the given string with specified radix. The range of the radix is 2 to 36. **Syntax:** public static Wrapper valueOf(String str, int radix);
3. **Wrapper valueOf(primitive p):** Every Wrapper class including Character class contains this method to create a Wrapper object for the given primitive type. **Syntax:** public static Wrapper valueOf(primitive p);

Methods of Wrapper Classes (Cont...)

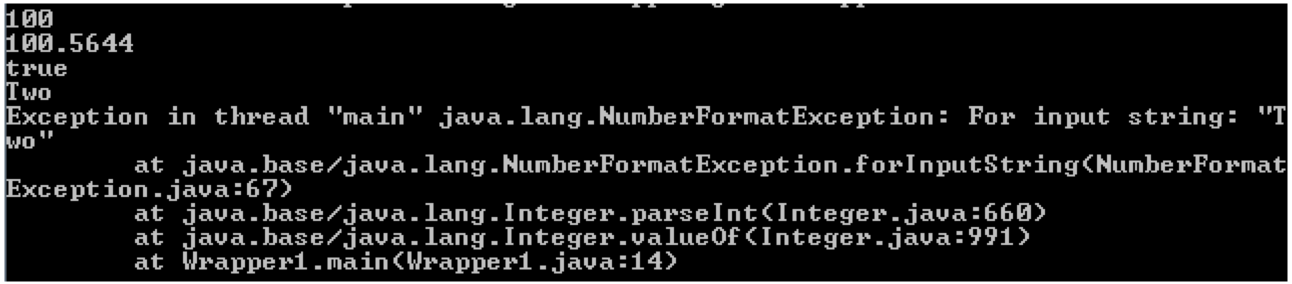
- **valueOf() method (Cont…)**



```
public class Wrapper1
{
   public static void main(String args[])
   {
      Integer I = Integer.valueOf("100");
      System.out.println(I);
      Double D = Double.valueOf("100.5644");
      System.out.println(D);
      Boolean B = Boolean.valueOf("true");
      System.out.println(B);
      String str = String.valueOf("Two");
      System.out.println(str);
      Integer I1 = Integer.valueOf("Two");
      System.out.println(I1);
  }
}
```

Methods of Wrapper Classes (Cont...)

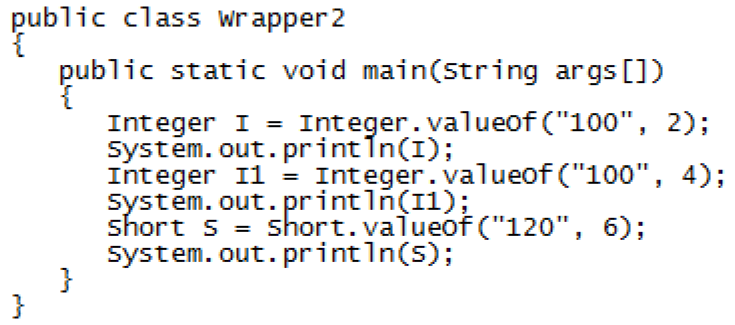
- valueOf() method (Cont…)
  - Output



```
100
100.5644
true
Two
Exception in thread "main" java.lang.NumberFormatException: For input string: "T
wo"
        at java.base/java.lang.NumberFormatException.forInputString(NumberFormat
Exception.java:67)
        at java.base/java.lang.Integer.parseInt(Integer.java:660)
        at java.base/java.lang.Integer.valueOf(Integer.java:991)
        at Wrapper1.main(Wrapper1.java:14)
```

Methods of Wrapper Classes (Cont...)

- **valueOf() method (Cont…)**



```
public class Wrapper2
{
   public static void main(String args[])
   {
      Integer I = Integer.valueOf("100", 2);
      System.out.println(I);
      Integer I1 = Integer.valueOf("100", 4);
      System.out.println(I1);
      Short S = Short.valueOf("120", 6);
      System.out.println(S);
   }
}
```

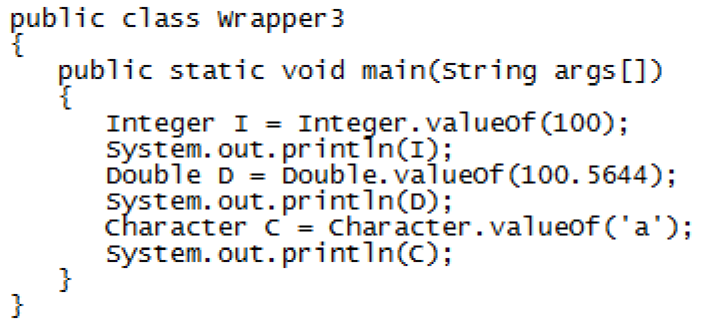
Methods of Wrapper Classes (Cont...)

- **valueOf() method (Cont…)**
  - **Output**

```
4
16
48
```

Methods of Wrapper Classes (Cont...)

- **valueOf() method (Cont…)**



```
public class Wrapper3
{
    public static void main(String args[])
    {
        Integer I = Integer.valueOf(100);
        System.out.println(I);
        Double D = Double.valueOf(100.5644);
        System.out.println(D);
        Character C = Character.valueOf('a');
        System.out.println(C);
    }
}
```

Methods of Wrapper Classes (Cont...)

- **valueOf() method (Cont…)**
  - **Output**

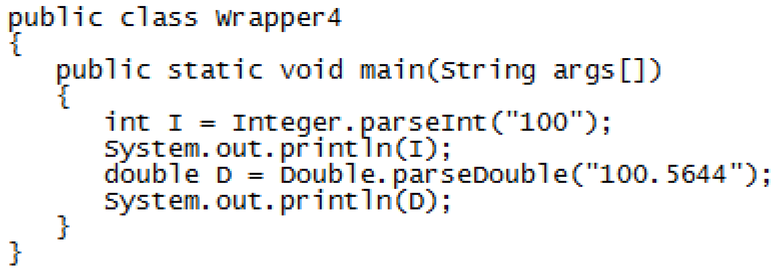
```
100
100.5644
a
```

Methods of Wrapper Classes (Cont…)

- **parseXxx() method**
  - We can use parseXxx() methods to convert string to primitive.
  - It is also a static method.
  - It returns Xxx type value.
  - It has two forms.
    1. **primitive parseXxx(String str):** Every Wrapper class except character class contains parseXxx() method to get primitive for the given string object. **Syntax:** public static primitive parseXxx(String str);
    2. **parseXxx(String str, int radix):** Every integral type Wrapper class (Byte, Short, Integer and Long) contains this method to convert specified radix string to primitive. **Syntax:** public static primitive parseXxx(String str, int radix);

Methods of Wrapper Classes (Cont...)

- **parseXxx() method (Cont…)**



```
public class Wrapper4
{
   public static void main(String args[])
   {
      int I = Integer.parseInt("100");
      System.out.println(I);
      double D = Double.parseDouble("100.5644");
      System.out.println(D);
   }
}
```

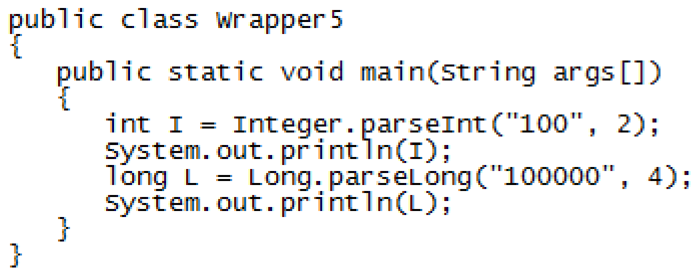
Methods of Wrapper Classes (Cont...)

- **parseXxx() method (Cont…)**
  - **Output**

```
100
100.5644
```

Methods of Wrapper Classes (Cont...)

- **parseXxx() method (Cont…)**



```
public class Wrapper5
{
    public static void main(String args[])
    {
        int I = Integer.parseInt("100", 2);
        System.out.println(I);
        long L = Long.parseLong("100000", 4);
        System.out.println(L);
    }
}
```

Methods of Wrapper Classes (Cont...)

- **parseXxx() method (Cont…)**
  - **Output**

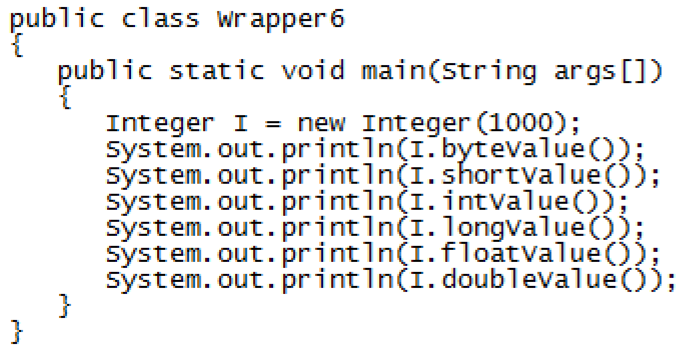
```
4
1024
```

Methods of Wrapper Classes (Cont...)

- **xxxValue() method**
  - We can use xxxValue() methods to get the primitive for the given Wrapper object.
  - It returns xxx primitive type.
  - Every number type Wrapper class (Byte, Short, Integer, Long, Float, Double) contains the following 6 methods to get primitive for the given Wrapper object:
    1. public byte byteValue()
    2. public short shortValue()
    3. public int intValue()
    4. public long longValue()
    5. public float floatValue()
    6. public float doubleValue()

Methods of Wrapper Classes (Cont…)

- **xxxValue() method (Cont…)**



```
public class Wrapper6
{
    public static void main(String args[])
    {
        Integer I = new Integer(1000);
        System.out.println(I.byteValue());
        System.out.println(I.shortValue());
        System.out.println(I.intValue());
        System.out.println(I.longValue());
        System.out.println(I.floatValue());
        System.out.println(I.doubleValue());
    }
}
```

Methods of Wrapper Classes (Cont...)

- **xxxValue() method (Cont…)**
  - **Output**

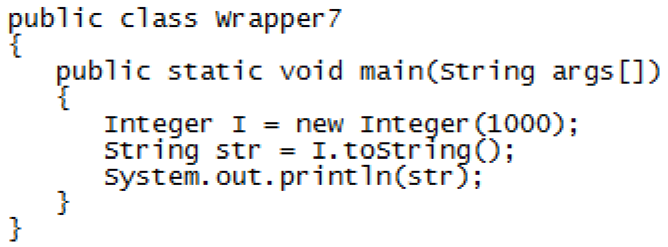
```
-24
1000
1000
1000
1000.0
1000.0
```

Methods of Wrapper Classes (Cont...)

- **toString() method**
  - This method is used to convert Wrapper object or primitive to string.
  - It has three forms.
    1. **toString():** Every wrapper class contains the toString() method to convert Wrapper object to string type. **Syntax:** public String toString();
    2. **toString(primitive p):** Every Wrapper class including Character class contains this static toString() method to convert primitive to string. **Syntax:** public static String toString(primitive p);
    3. **toString(primitive p, int radix):** Integer and Long classes contains this type of toString() method to convert primitive to specified radix string. **Syntax:** public static String toString(primitive p, int radix);

Methods of Wrapper Classes (Cont...)

- **toString() method (Cont…)**



```
public class Wrapper7
{
   public static void main(String args[])
   {
      Integer I = new Integer(1000);
      String str = I.toString();
      System.out.println(str);
   }
}
```

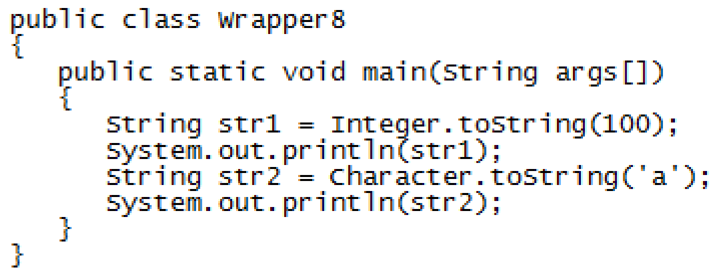
Methods of Wrapper Classes (Cont…)

- **toString() method (Cont…)**
  - **Output**

```
100
```

Methods of Wrapper Classes (Cont…)

- **toString() method (Cont…)**



```
public class Wrapper8
{
   public static void main(String args[])
   {
      String str1 = Integer.toString(100);
      System.out.println(str1);
      String str2 = Character.toString('a');
      System.out.println(str2);
   }
}
```

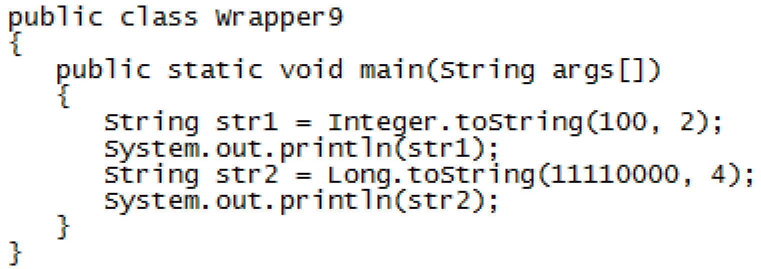
Methods of Wrapper Classes (Cont...)

- **toString() method (Cont…)**
  - **Output**

```
100
a
```

Methods of Wrapper Classes (Cont...)

- **toString() method (Cont…)**



```
public class Wrapper9
{
    public static void main(String args[])
    {
        String str1 = Integer.toString(100, 2);
        System.out.println(str1);
        String str2 = Long.toString(11110000, 4);
        System.out.println(str2);
    }
}
```

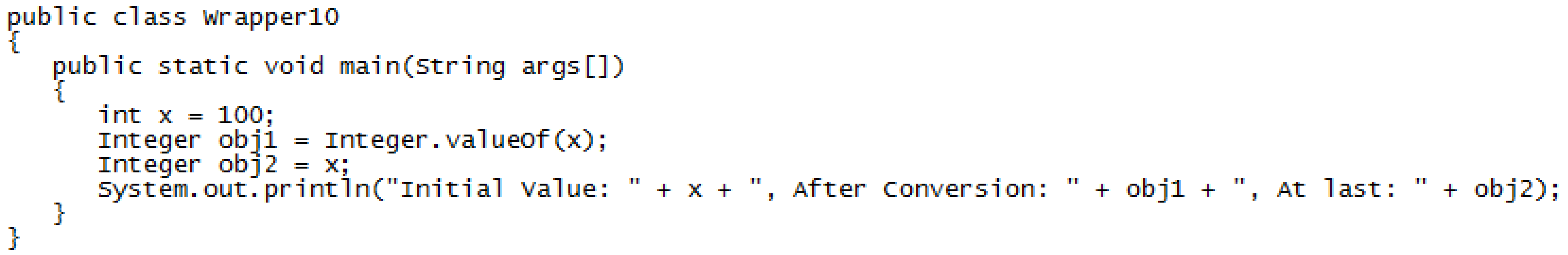
Methods of Wrapper Classes (Cont...)

- **toString() method (Cont…)**
  - **Output**

```
1100100
222120121300
```

Autoboxing and Unboxing

- **Autoboxing**
  - Converting primitive data type to wrapper type.



```
public class Wrapper10
{
   public static void main(String args[])
   {
      int x = 100;
      Integer obj1 = Integer.valueOf(x);
      Integer obj2 = x;
      System.out.println("Initial Value: " + x + ", After Conversion: " + obj1 + ", At last: " + obj2);
   }
}
```

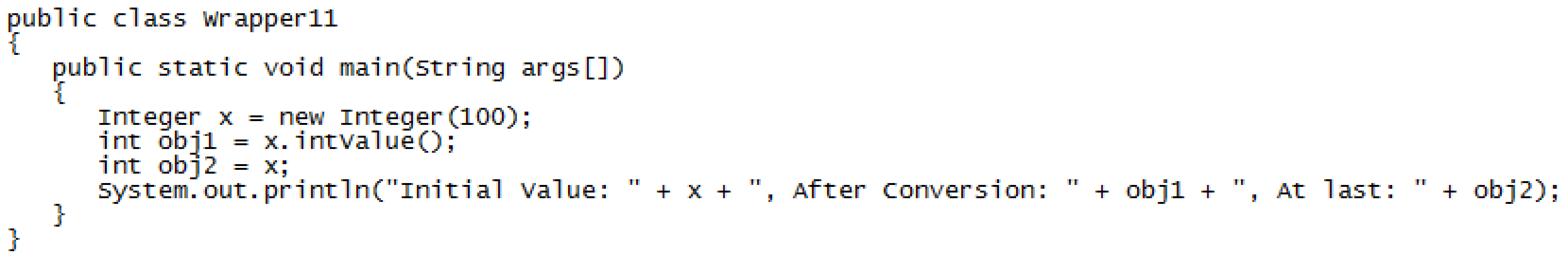
Autoboxing and Unboxing (Cont…)

- **Autoboxing (Cont…)**
  - Output

```
Initial Value: 100, After Conversion: 100, At last: 100
```

Autoboxing and Unboxing (Cont…)

- **Unboxing**
  - Converting wrapper type to respective primitive data type.
  - It is the reverse process of autoboxing.



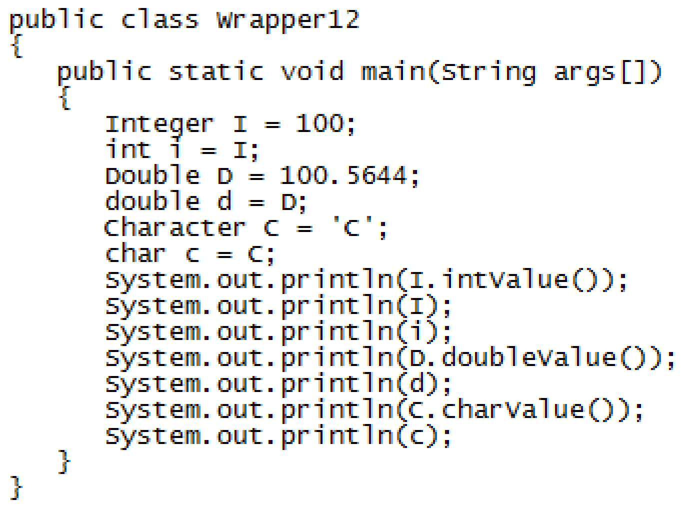
```
public class Wrapper11
{
    public static void main(String args[])
    {
        Integer x = new Integer(100);
        int obj1 = x.intValue();
        int obj2 = x;
        System.out.println("Initial Value: " + x + ", After Conversion: " + obj1 + ", At last: " + obj2);
    }
}
```

Autoboxing and Unboxing (Cont…)

- **Unboxing (Cont…)**
  - Output

```
Initial Value: 100, After Conversion: 100, At last: 100
```

Autoboxing and Unboxing (Cont…)



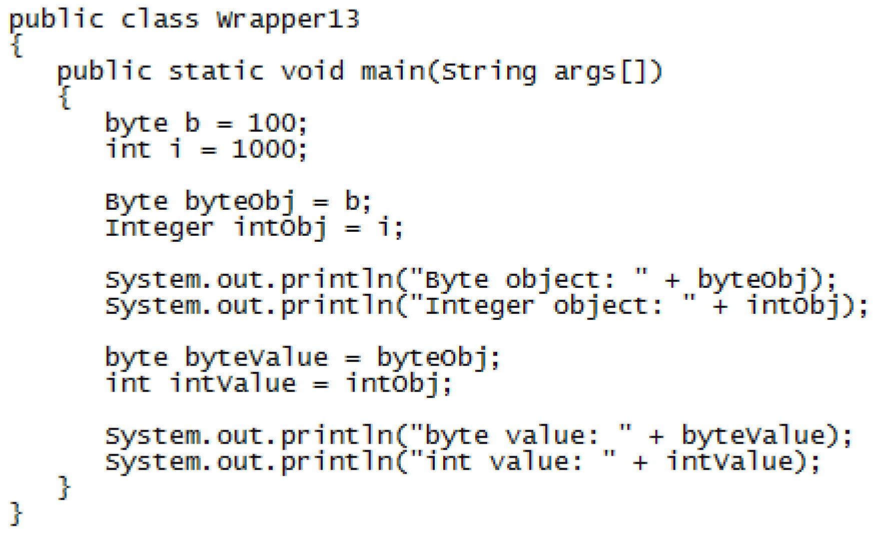
```
public class Wrapper12
{
    public static void main(String args[])
    {
        Integer I = 100;
        int i = I;
        Double D = 100.5644;
        double d = D;
        Character C = 'C';
        char c = C;
        System.out.println(I.intValue());
        System.out.println(I);
        System.out.println(i);
        System.out.println(D.doubleValue());
        System.out.println(d);
        System.out.println(C.charValue());
        System.out.println(c);
    }
}
```

Autoboxing and Unboxing (Cont…)

- **Unboxing (Cont…):**
  - Output

```
100
100
100
100.5644
100.5644
C
C
```

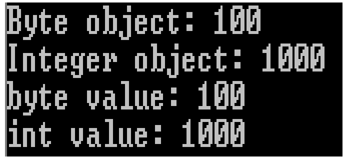
Autoboxing and Unboxing (Cont…)



```
public class Wrapper13
{
    public static void main(String args[])
    {
        byte b = 100;
        int i = 1000;

        Byte byteObj = b;
        Integer intObj = i;

        System.out.println("Byte object: " + byteObj);
        System.out.println("Integer object: " + intObj);

        byte byteValue = byteObj;
        int intValue = intObj;

        System.out.println("byte value: " + byteValue);
        System.out.println("int value: " + intValue);
    }
}
```

Autoboxing and Unboxing (Cont…)

- **Unboxing (Cont…):**
  - Output



```
Byte object: 100
Integer object: 1000
byte value: 100
int value: 1000
```

thank
you . . .

Slides are prepared from various sources, such as Book, Internet Links and many more.