Class Library
java.lang Package

Bok, Jong Soon
javaexpert@nate.com
https://github.com/swacademy/Core-Java

Object class

- Is the root of the class hierarchy.
- Every class has Object as a superclass.
- If no inheritance is specified when a class is defined, the superclass of the class is Object by default.

Object
<<create>>+Object()
#clone(): Object
+equals(obj: Object): boolean
#finalize(): void
+getClass(): Class<?>
+hashCode(): int
+notify(): void
+notifyAll(): void
+toString(): String
+wait(): void
+wait(timeout: long): void
+wait(timeout: long, nanos: int): void

Object class – clone()

■ Performs a "shallow copy" of this object, not a "deep copy" operation.

```java
1  public class CloneTest implements Cloneable{
2      private String name;
3      public CloneTest(String name) { this.name = name; }
4      public String getName() { return this.name; }
5      public void setName(String name) { this.name = name; }
6      public static void main(String[] args) {
7          CloneTest  t = new CloneTest("Duncan");
8          Object obj = null;
9          try{
10             obj = t.clone();
11         }catch(CloneNotSupportedException ex){
12             System.out.println("Cannot Copy");
13         }
14         CloneTest  t1 = (CloneTest)obj;
15         if(t == t1) System.out.println("Equals");
16         else System.out.println("Not Equals");
17         System.out.println(t1.getName());
18         t.setName("Michael");
19         System.out.println(t1.getName());
20     }
21 }
```

```
---------- Java Interpreter ----------
Not Equals
Duncan
Duncan
```

Object class – clone() (Cont.)

- To doing "deep copy" using method's *override*.

```java
2 public class CloneTest {
3   public static void main(String[] args) {
4       // TODO Auto-generated method stub
5       Product original = new Product(10);
6       Object obj = original.clone();
7       if(obj instanceof Product){
8           Product other = (Product)obj;
9           System.out.println(other.getSu());
10      }
11  }
12 }
13 class Product {
14     private int su;
15     public Product(int su){
16         this.su = su;
17     }
18     protected Object clone(){ return this; }
19     public int getSu() { return this.su; }
20 }
```

Object class – equals()

```java
2  public class EqualsTest {
3      public static void main(String[] args) {
4          // TODO Auto-generated method stub
5          Person original = new Person("Michael");
6          Person other = new Person("Sujan");
7          if(original.equals(other))
8              System.out.println("Name is same.");
9          else
10             System.out.println("Name is different.");
11     }
12 }
13 class Person{
14     private String name;
15     public Person(String name) { this.name = name; }
16     public boolean equals(Object obj){  //Object's equals() override
17         Person other = (Person)obj;
18         if(other.name == this.name) return true;
19         else return false;
20     }
21 }
```

Object class – finalize()

- Called by the garbage collector on an object when garbage collection determines that there are no more references to the object.

- A subclass overrides the `finalize` method to dispose of System resources or to perform other cleanup.

- Is never invoked more than once by a Java virtual machine.

Object class – finalize() (Cont.)

```java
1  //finalize()
2  public class ObjectDemo {
3      public static void main(String[] args) {
4          Demo d = new Demo();
5          d = null;
6          System.gc();
7      }
8  }
9  class Demo{
10     public Demo(){
11         System.out.println("Demo class's Constructor");
12     }
13     @Override
14     protected void finalize(){
15         System.out.println("Demo class's Deconstructor");
16     }
17 }
```

```
Demo class's Constructor
Demo class's Deconstructor
```

Object class – getClass()

- Returns the runtime class of an object.

```java
2  public class GetClassTest {
3      public static void main(String[] args) {
4          // TODO Auto-generated method stub
5          Cat c = new Cat();
6          System.out.println(c.getClass());
7      }
8  }
9  class Cat{}
10 /*
11 class Cat
12 */
```

Object class – toString()

```
 2  public class ToStringTest {
 3      public static void main(String[] args) {
 4          // TODO Auto-generated method stub
 5          Dog d = new Dog();
 6          d.display();
 7          System.out.println(d.toString());
 8      }
 9  }
10  class Dog{
11      public void display(){
12          System.out.println(this);
13      }
14  }
15  /*
16   * Dog@757aef
17   * Dog@757aef
18   */
```

```
 2  public class ToStringTest {
 3      public static void main(String[] args) {
 4          // TODO Auto-generated method stub
 5          Dog d = new Dog("Michael");
 6          System.out.println(d.toString());
 7      }
 8  }
 9  class Dog{
10      private String name;
11      public Dog(String name){ this.name = name; }
12      public String toString(){
13          return String.format("This Dog's name is %s...", this.name);
14      }
15  }
16  /*
17   * This Dog's name is Michael...
18   */
```

Wrapper Classes

- Look at primitive data elements as objects.

Primitive Data Type	Wrapper Class	Methods to get primitive values
boolean	Boolean	booleanValue()
byte	Byte	charValue()
char	Character	byteValue()
short	Short	shortValue()
int	Integer	intValue()
long	Long	longValue()
float	Float	floatValue()
double	Double	doubleValue()

Wrapper Classes (Cont.)
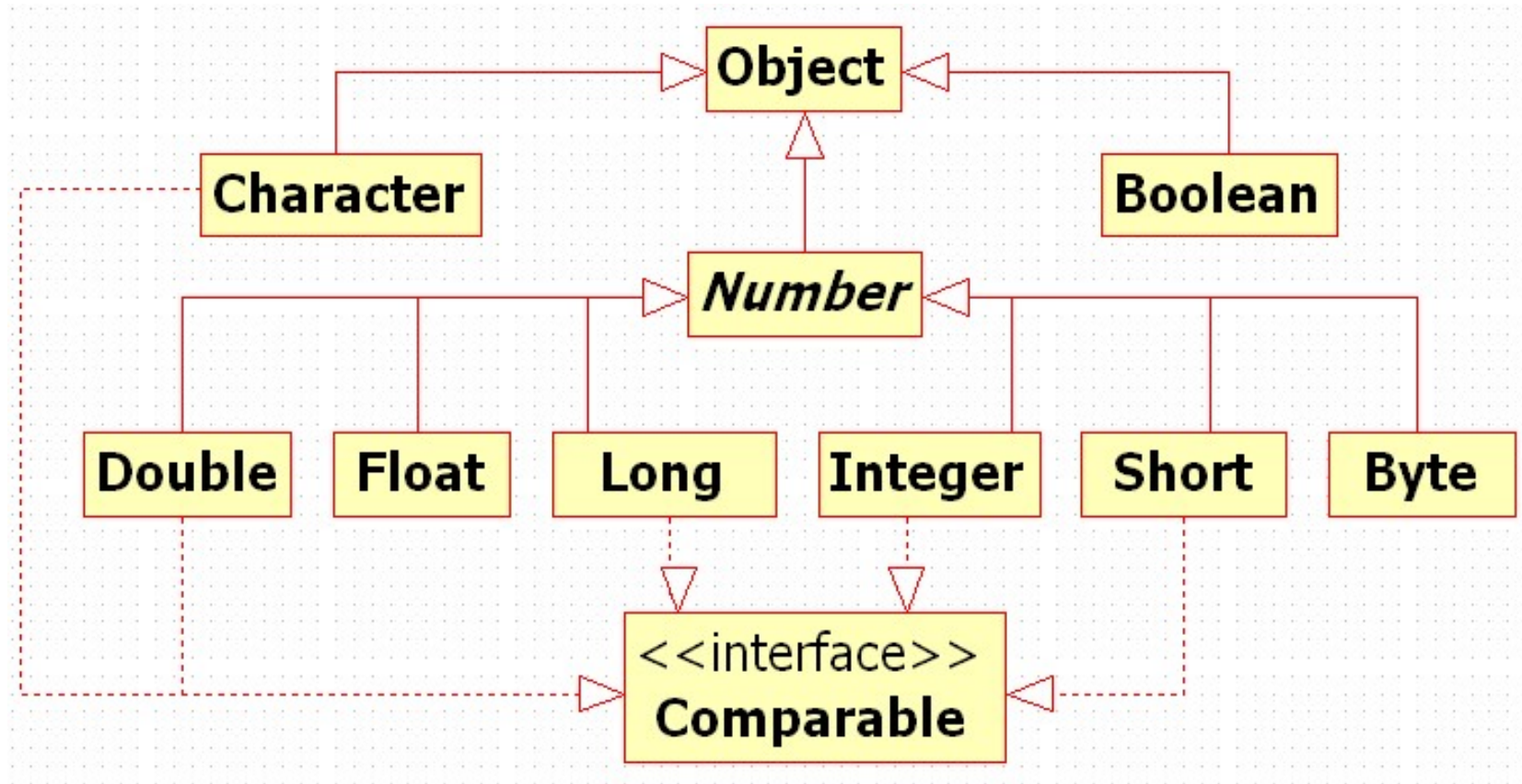
Boolean Class

- Wraps a value of the primitive type `boolean` in an object.

java.lang.Boolean
+FALSE: Boolean
+TRUE: Boolean
<<create>>+Boolean(value: boolean)
<<create>>+Boolean(s: String)
+booleanValue(): boolean
+compareTo(b: java.lang.Boolean): int
+getBoolean(name: String): boolean
+parseBoolean(s: String): boolean
+toString(b: boolean): String
+valueOf(b: boolean): Boolean
+valueOf(s: String): Boolean

Numeric Wrapper Classes

- Each wrapper class overrides the `toString()`, `equals()`, and `hashCode()`.

- All the numeric wrapper classes and the `Character` class implement the `Comparable` interface → the `compareTo()` is implemented in these classes.

Numeric Wrapper Classes
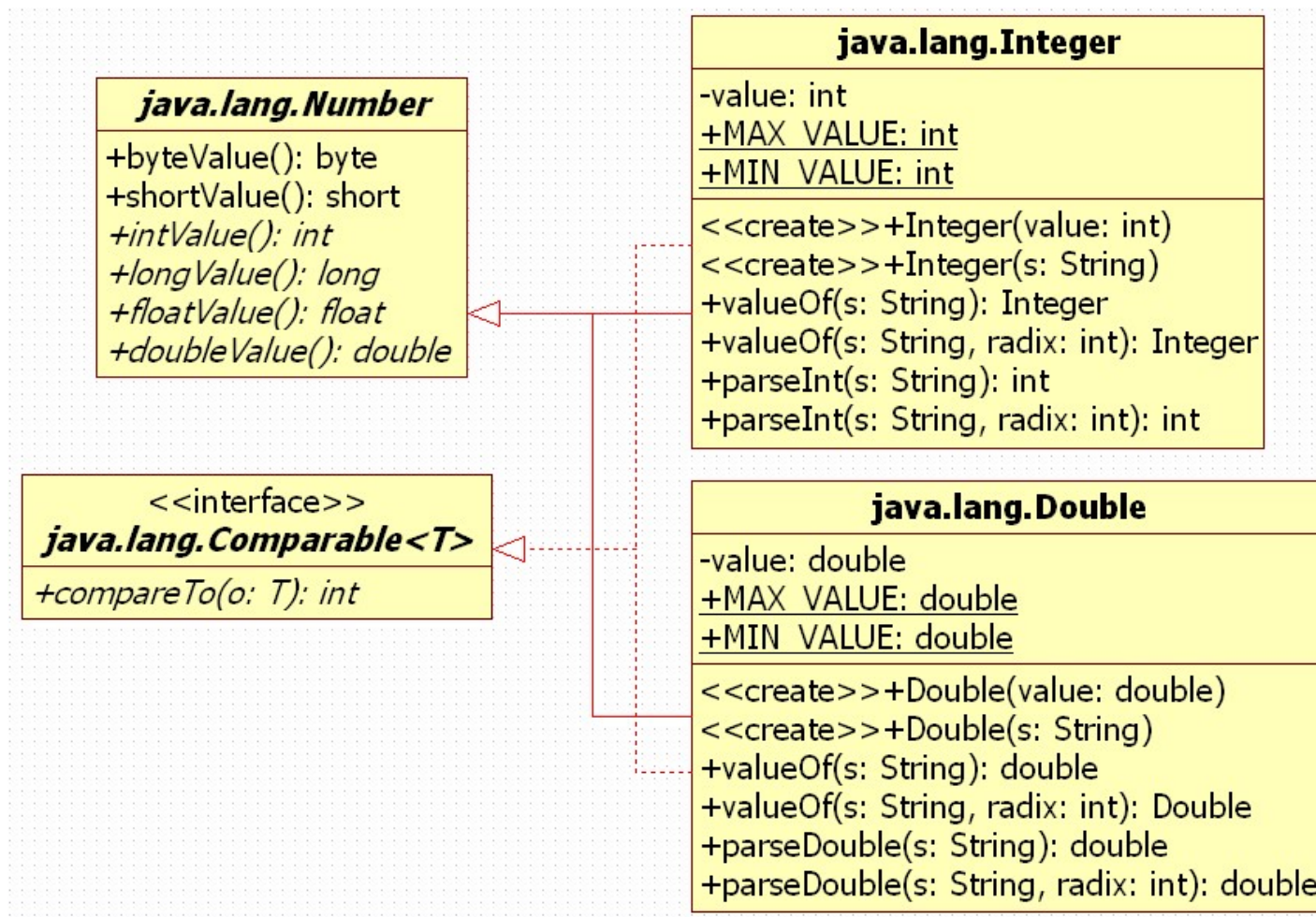
Numeric Wrapper Classes

- Common Constructors
  - `Datatype (dataType value)`
  - `Datatype (String s) throws NumberFormatException`
- Common Fields
  - `static dataType MAX_VALUE`
  - `static dataType MIN_VALUE`

Numeric Wrapper Classes (Cont.)

■ Common Methods

- `dataType datatypeValue ()`
- `dataType compareTo (Datatype anotherDataType)`
- `static dataType parseDataType (String s)`
- `boolean equals (Object obj)`
- `String toString()`
- `static Datatype valueOf (String s)`

Numeric Wrapper Classes (Cont.)

- Etc methods
  - `static String toBinaryString (int i)`
  - `static String toHexString (int i)`
  - `static String toOctalString (int i)`

`Character` Class

- Wraps a value of the primitive type char in an object.
- Character information is based on the Unicode Standard, version 4.0

Character Class

- Wraps a value of the primitive type char in an object.
- Character information is based on the Unicode Standard, version 4.0
- `Character (char value)`
- `static boolean isXxx (char ch)`
- `static boolean isXxx (int codePoint)`

Autoboxing and Autounboxing

```java
public class AutoboxingTest {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Integer in = new Integer(10);
        int su = 5;
        su += in;               //autounboxing
        in = su;                //autoboxing
        System.out.println("in = " + in);
    }
}
/*
 * in = 15
 */
```

Math Class

- Contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.

- Is final class. So, Cannot inherit.

- All field is final fields. So, Cannot assign literal.

- All method is final method. So, Cannot override it.

Math class (Cont.)

- `static type abs (type value)`
- `static type max (type a, type b)`
- `static type min (type a, type b)`
- `static double pow (double a, double b)`
- `static double sqrt (double a)`
- `static double random ()`
- `static double ceil (double a)`
- `static double floor (double a)`
- `static int round (float a)`
- `static long round (double a)`

`String` class

- Represents character strings.
- All string literals are implemented as instances of this class.
- Strings are *constant*.
- Their values cannot be changed after they are created.
- Case mapping relies heavily on the information provided by the Unicode Consortium's Unicode 4.0 specification.

String class (Cont.)

- Provides special support for the string concatenation operator (**+**), and for conversion of other objects to strings.
- String concatenation is implemented through the **StringBuffer** class and its append method.

String class (Cont.)

- `String (byte [] bytes)`
- `String (byte [] bytes, int offset, int length)`
- `String (byte [] bytes, int offset, int length, String charsetName)`
- `String (byte [] bytes, String charsetName)`
- `String (char [] value)`
- `String (char [] value, int offset, int count)`
- `String (String original)`
- `String (StringBuffer buffer)`

String class (Cont.)

- char charAt (int index)
- int compareTo (String anotherString)
- int compareToIgnoreCase (String str)
- String concat (String str)
- static String copyValueOf (char [] data)
- static String copyValueOf (char [] data, int offset, int count)
- boolean endsWith (String suffix)
- boolean equals (Object anObject)
- boolean equalsIgnoreCase (String anotherString)

String class (Cont.)

- `byte [] getBytes ()`
- `byte [] getBytes (String charsetName)`
- `void getChars (int srcBegin, int srcEnd, char [] dst, int dstBegin)`
- `int hashCode ()`
- `int indexOf (int ch)`
- `int indexOf (int ch, int fromIndex)`
- `int indexOf (String str)`
- `int indexOf (String str, int fromIndex)`

String class (Cont.)

- `int lastIndexOf (int ch)`
- `int lastIndexOf (String str)`
- `int length ()`
- `boolean matches (String regex)`
- `boolean regionMatches (boolean ignoreCase, int toffset, String other, int ooffset, int len)`
- `String replace (char oldChar, char newChar)`
- `String replaceAll (String regex, String replacement)`
- `boolean startsWith (String prefix)`

String class (Cont.)

- `String substring (int beginIndex)`
- `String substring (int beginIndex, int endIndex)`
- `char [] toCharArray ()`
- `String toLowerCase ()`
- `String toString()`
- `String toUpperCase ()`
- `String trim ()`
- `static String valueOf (type t)`

Unicode Character Set(UCS)*

Charset	Description
US-ASCII	Seven-bit ASCII, ISO646-US, the Basic Latin block of the Unicode character set
ISO-8859-1	ISO Latin Alphabet No. 1, ISO-LATIN-1
UTF-8	Eight-bit UCS Transformation Format
UTF-16BE	Sixteen-bit UCS Transformation Format, big-endian byte order
UTF-16LE	Sixteen-bit UCS Transformation Format, little-endian byte order
UTF-16	Sixteen-bit UCS Transformation Format, byte order identified by an optional byte-order mark
EUC**-KR	Eight-bit, Unified Hangeul Code (통합완성형)
KS C 5601	KS_C_5601-1987, KSC_5601, KSC5601, KS X 1001, Korean Graphic Character Set for Information Interchange
MS949***	Code page 949, Windows Codepage 949, Microsoft's implementation that appears similar to EUC-KR

*   java.nio.charset.Charset

*   Extended Unix Code (EUC) is a multibyte character encoding system used primarily for Korean, Japanese and simplified Chinese.

**   http://msdn.microsoft.com/ko-kr/goglobal/cc305154

`String` class (Cont.)

```java
2 public class StringDemo1 {
3    public static void main(String[] args) {
4        // TODO Auto-generated method stub
5        char [] array = {'H','e','l','l','o'};
6        String str = new String(array);
7        String str1 = new String(str);
8        System.out.println("str = " + str);
9        System.out.println("str1 = " + str1);
10   }
11 }
12 /*
13 str = Hello
14 str1 = Hello
15 */
```

String class (Cont.)

```java
byte [] array = "안녕하세요".getBytes();
for(byte b : array){
    System.out.print(b + ", ");
}
```

-20, -107, -120, -21, -123, -107, -19, -107, -104, -20, -124, -72, -20, -102, -108,

String class (Cont.)

```java
byte [] array = {65, 66, 67, 68, 69, 70};
String str = new String(array);
System.out.println("str = " + str);

byte [] array1 = {-20, -107, -120, -21, -123, -107, -19, -107,
                  -104, -20, -124, -72, -20, -102, -108};
String str1 = new String(array1, "ISO8859_1");
String str2 = new String(array1, "KSC5601");
String str3 = new String(array1, "EUC-KR");
String str4 = new String(array1, "UTF-8");
String str5 = new String(array1, "UTF-16");

System.out.println("str1 = " + str1);
System.out.println("str2 = " + str2);
System.out.println("str3 = " + str3);
System.out.println("str4 = " + str4);
System.out.println("str5 = " + str5);
```

```
str = ABCDEF
str1 = ì ë í ì„¸
str2 = ▯▯▯▯▯▯
str3 = ▯▯▯▯▯▯
str4 = 안녕하세요
str5 =    襧薕   養蒸   ▯
```

String class (Cont.)

```java
2  public class StringDemo {
3      public static void main(String[] args) {
4          // TODO Auto-generated method stub
5          String str = "C";
6          String str1 = "A";
7          int su = str.compareTo(str1);
8          System.out.println("su = " + su);
9      }
10 }
11 /*
12 su = 2
13 */
```

```java
2  public class StringDemo3 {
3      public static void main(String[] args) {
4          // TODO 자동 생성된 메소드 스텁
5          String str = "Hello, World";
6          String str1 = "안녕 하세요";
7          System.out.println(str.length());
8          System.out.println(str1.length());
9      }
10 }
11 /*
12 12
13 7
14 */
```

`String` class (Cont.)

```java
/*String str = "Hello";
String str1 = "Hello";*/

String str = new String("Hello");
String str1 = new String("Hello");

if(str.contentEquals(str1)) System.out.println("Equals");
else System.out.println("Different");

//cf. boolean equals(Object anObject), boolean equalsIgnoreCase(Object anotherString)
//cf. int compareTo(String anotherStr), int compareToIgnoreCase(String str)
```

String class (Cont.)

```java
2  public class StringDemo3 {
3      public static void main(String[] args) {
4          // TODO 자동 생성된 메소드 스텁
5          String msg = "446-912 경기도 용인시 기흥구 마북동 431번지";
6          String zip1 = msg.substring(0, 3);
7          String zip2 = msg.substring(4, 7);
8          String address= msg.substring(8);
9          System.out.println("우편번호 = " + zip1+ "-" + zip2);
10         System.out.println("주소 = " + address);
11     }
12 }
13 /*
14 우편번호 = 446-912
15 주소 = 경기도 용인시 기흥구 마북동 431번지
16 */
```

`String` class (Cont.)

```java
2 public class StringDemo3 {
3     public static void main(String[] args) {
4         // TODO Auto-generated method stub
5         String str = "Java is a Object Oriented Programming";
6
7         System.out.println(str.indexOf("Java"));
8         System.out.println(str.lastIndexOf("a"));
9         System.out.println(str.indexOf('a', 5));
10
11         String msg = "Hello";
12         //msg[0] = 'C';  //error
13         msg = msg.replace('H', 'C');
14         System.out.println("msg = " + msg);
15     }
16 }
17 /*
18 0
19 31
20 8
21 msg = Cello
22 */
```

`StringBuffer` class

- Is final class.
- A string buffer implements a *mutable* sequence of characters. A string buffer is like a String, but can be modified.
- String buffers are used by the compiler to implement the binary string concatenation operator **+**.
- Every string buffer has a capacity. As long as the length of the character sequence contained in the string buffer does not exceed the capacity, it is not necessary to allocate a new internal buffer array. If the internal buffer overflows, it is automatically made larger.

StringBuffer class (Cont.)

- StringBuffer (int length)
- StringBuffer (String str)
- StringBuffer append (type t)
- int capacity ()
- StringBuffer delete (int start, int end)
- StringBuffer deleteCharAt (int index)
- void getChars (int srcBegin, int srcEnd, char [] dst, int dstBegin)
- int indexOf(String str)

StringBuffer class (Cont.)

- StringBuffer insert (int offset, type t)
- int lastIndexOf (String str)
- int length ()
- StringBuffer replace (int start, int end, String str)
- StringBuffer reverse ()
- void setCharAt (int index, char ch)
- void setLength (int newLength)
- String substring (int start)
- String substring (int start, int end)

StringBuffer class (Cont.)

```java
2 public class StringBufferDemo {
3     public static void main(String[] args) {
4         // TODO Auto-generated method stub
5         StringBuffer sb = new StringBuffer();
6         System.out.println(sb.capacity());
7         StringBuffer sb1 = new StringBuffer(30);
8         System.out.println(sb1.capacity());
9         StringBuffer sb2 = new StringBuffer("Hello");
10        System.out.println("buffer = " + sb2);
11        System.out.println("length = " + sb2.length());
12        System.out.println("capacity = " + sb2.capacity());
13    }
14 }
15 /*
16 16
17 30
18 buffer = Hello
19 length = 5
20 capacity = 21
21 */
```

StringBuffer class (Cont.)

```java
 2 public class StringBufferDemo {
 3     public static void main(String[] args) {
 4         // TODO Auto-generated method stub
 5         String str = new String("A");
 6         long startString = System.currentTimeMillis();
 7         for(int i = 0 ; i < 50000 ; i++){
 8             str = str.concat("A");
 9         }
10         long endString = System.currentTimeMillis();
11         StringBuffer sb = new StringBuffer("A");
12         long startStringBuffer = System.currentTimeMillis();
13         for(int i = 0 ; i < 50000 ; i++){
14             sb.append("A");
15         }
16         long endStringBuffer = System.currentTimeMillis();
17         System.out.println("String : " + (endString - startString) + "ms");
18         System.out.println("StringBuffer : " + (endStringBuffer - startStringBuffer) + "ms");
19     }
20 }
21 /*
22 String : 3985ms
23 StringBuffer : 15ms
24 */
```

StringBuffer class (Cont.)

```java
2 public class StringBufferDemo {
3     public static void main(String[] args) {
4         // TODO Auto-generated method stub
5         StringBuffer sb = new StringBuffer("I Java!");
6         sb.insert(2, "love ");   System.out.println(sb);
7         sb.reverse();   System.out.println(sb);
8         sb.reverse();
9         sb.delete(7, 11);   System.out.println(sb);
10        sb.deleteCharAt(0);   System.out.println(sb);
11        sb.insert(6, "Java");   System.out.println(sb);
12        sb.replace(6, 10, "XML");   System.out.println(sb);
13    }
14 }
15 /*
16 I love Java!
17 !avaJ evol I
18 I love !
19  love !
20  love Java!
21  love XML!
22 */
```

`System` **class**

- Is final class.

- Contains several useful class fields and methods. It cannot be instantiated.

- Among the facilities provided by the System class are standard input, standard output, and error output streams; access to externally defined "properties"

- `static PrintStream err` : Standard error output

- `static InputStream in` : Standard input

- `static PrintStream out` : Standard output

System class (Cont.)

- `static void arraycopy (Object src, int srcPos, Object dest, int destPos, int length)`
- `static long currentTimeMillis ()`
- `static void exit (int status)`
- `static void gc ()`
- `static Properties getProperties ()`
- `static String getProperty (String key)`
- `static String getProperty (String key, String def)`

System class (Cont.)

```java
Map<String, String> map = System.getenv();
Set<String> keys = map.keySet();
Object [] array = keys.toArray();
for(Object key : array){
    System.out.println(key + "=" + map.get(key));
}
```

```
PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.PSC1
JAVA_HOME=C:\Program Files\Java\jdk1.7.0_25
TEMP=C:\DOCUME~1\Redmond\LOCALS~1\Temp
SystemDrive=C:
MOZ_PLUGIN_PATH=C:\Program Files\Foxit Software\Foxit Reader\plugins\
ProgramFiles=C:\Program Files
Path=C:/Program Files/Java/jdk1.7.0_25/bin/../jre/bin/client;C:/Program Fi
```

System class (Cont.)

```java
long startMili = System.currentTimeMillis();
long startNano = System.nanoTime();
for(int i = 0 ; i < Integer.MAX_VALUE ; i++);
long endMili = System.currentTimeMillis();
long endNano = System.nanoTime();

System.out.println("Difference Milisec = " + (endMili - startMili) + "ms");
System.out.println("Difference Nanosec = " + (endNano - startNano) + "ns");
```

```
Difference Milisec = 2391ms
Difference Nanosec = 2379537064ns
```

`Runtime` class

- Every Java application has a single instance of class Runtime.
- Allows the application to interface with the environment in which the application is running.
- The current runtime can be obtained from the `getRuntime()` method.
- An application cannot create its own instance of this class.

Runtime class (Cont.)

```java
5       Runtime r = Runtime.getRuntime();
6       long memory, memory1;
7       Integer [] array = new Integer[1000];
8       System.out.println("Total Memory : " + r.totalMemory());
9       memory = r.freeMemory();
10      System.out.println("Free Memory : " + memory);
11      r.gc();
12      memory = r.freeMemory();
13      System.out.println("After GC Free Memory : " + memory);
14      for(int i=0;i < array.length; i++)
15          array[i] = new Integer(i);
16      memory1 = r.freeMemory();
17      System.out.println("After fullfill Free Memory : " + memory1);
18      System.out.println("Memory that assigns data : " + (memory - memory1));
19      for(int i=0;i < array.length; i++)
20          array[i] = null;
21      r.gc();
22      memory1 = r.freeMemory();
23      System.out.println("After GC Free Memory : " + memory1);
24    }
25 }
26 /*
27 Total Memory : 2031616
28 Free Memory : 1836696
29 After GC Free Memory : 1892056
30 After fullfill Free Memory : 1875728
31 Memory that assigns data : 16328
32 After GC Free Memory : 1892056
33 */
```

Runtime class (Cont.)

```java
2  public class RuntimeDemo {
3      public static void main(String[] args) {
4          // TODO Auto-generated method stub
5          Runtime r = Runtime.getRuntime();
6          Process p = null;
7          try{
8              p = r.exec("Notepad");
9          }catch(Exception e){
10             System.out.println("Error executing Notepad.");
11         }
12     }
13 }
```