清森学校
BEIJING QINGSEN SCHOOL

AP-CSA
Using Objects

YING HUANG

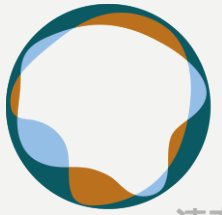
SEP.2022

STRING

REVIEW STRING

```
String name = "Ying";
```

- **String**: An object storing a sequence of text characters.
- **String** is not a primitive type. String is an object/ reference data type.



STRING ARE OBJECTS

Three ways to initialize a string:

1. String name = "…";
2. String name = new String ("…");
3. String name = expression;

The String class is part of the java.lang package. Classes in the java.lang package are available by default.

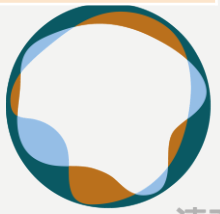
Example:

```
String name1 = "Ying Huang";
String name2 = new String("Ying Huang");
String name3 = "Ying"+" "+"Huang"
```

STRING METHODS

Since String are Objects, they also have Methods!

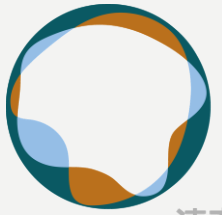
Method	Use
String concat(String str)	Return a new String with str appended to the end of string value
String replace(String str1,String str2)	Return a new String with all instances of str1 in string replaced with str2
String toUpperCase()	Returns a new String as Upper Case Letter



IMMUTABILITY 不可改变

- String methods do not alter the existing String, the create **new one**!

- The only way to change the value of name is to **re-assign** it!



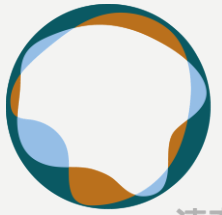
CONCATENATION 连接

- String can be concatenated with one another to create a new String value.

```
String firstNanme = "Ying";
String lastName = "Huang";
String full name = firstName+lastName;
```

- Use the += shortcut on String value as well

```
String name = "Ying";
name+="!" // name is Ying!
```



CONCATENATION 连接

- Primitive values can be concatenated with a String object using +. This causes implicit conversion of the values to String objects.

```
String num1 = "3";
String num2 = num1+4; //num2 = 34
```

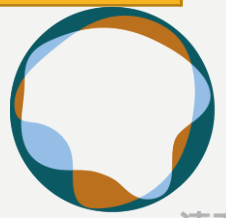
```
int num1 = 20;
int num2 = 22;
System.out.println("the sum is "+num1+num2);
//the sum is 2022
System.out.println("the sum is "+ (num1+num2));
//the sum is 42
```

CONCATENATION 连接

Be careful when concatenating Strings!

The + operator works from left to right!

```
int num1 = 20;
int num2 = 22;
System.out.println("the sum is "+num1+num2);
//the sum is 2022
System.out.println (num1+num2+ ("the sum is "+ ));
// 42 the sum is
```



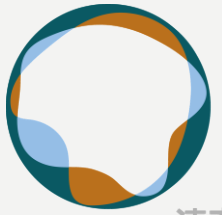
STRING DILEMMA

If we try to include quotes in a line of code, we will cause an error in our program.

```
String dialogue = "And he said, " Thank you!" ";
```

Escape sequence: A special sequence of characters used to represent certain special characters in a string.

Escape Sequence	Use
\n	new line character
\"	quotation mark character
\\	backslash character
\t	Tab space



EXERCISE 1

1. What is the output of the following `println` statements?

```
System.out.println("\\\\");
System.out.println("'");
System.out.println("\"\"\"");
System.out.println("This is a\t tab space")
```

2. Write a `println` statement to produce this output:

```
/ \ // \\ /// \\\
```

清森学校
BEIJING QINGSEN SCHOOL

EXERCISE 1 -KEY

1. Output of each `println` statement:

```
\\
'
"""
This is a	tab space
```

2. `println` statement to produce the line of output:

```
System.out.println("/ \\ // \\\\ /// \\\\\\");
```

清森学校
BEIJING QINGSEN SCHOOL

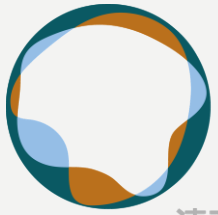
INDEXES

Characters of a string are numbered with 0-based indexes:

```
String name = "Ying Huang";
```

Index	0	1	2	3	4	5	6	7	8	9
Character	Y	i	n	g		H	u	a	n	g

- String's length: 10
- First character's index : 0
- Last character's index : 9 (1 less than the string's length)



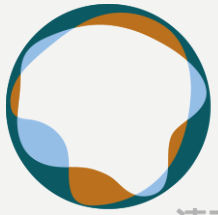
METHOD - INDEXES

Method	Use
int length()	returns the number of characters in the string
String substring(int index1, int index2)	Returns the characters in this string from index1 (inclusive) to index2 (exclusive);
String substring(int index2)	Returns the characters in this string from index1 (inclusive) to the end of string;
int indexOf(string str1)	Returns the index of the beginning of str in the current string or -1 if it isn't found.
int compareTo(String str2)	Returns a positive number is greater than str2; Return a negative number is less than str2; Return 0, it is equal to str2;
boolean equals(String str2)	Returns true if this is equal to other; Returns false otherwise

EXAMPLE

```
String message1 = "This is a test";
String message2 = "Hello Class";

int len1 = message1.length();
System.out.println(len1);// 14
int len2 = message2.length();
System.out.println(len2); // 11

String sub1 = message1.substring(0,4);
System.out.println(sub1); // This
String sub2 = message1.substring(5);
System.out.println(sub2); //is a test
```

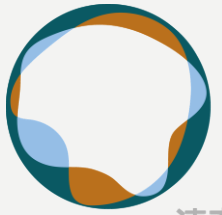
```
int index1 = message1.indexOf("is”);
System.out.println(index1); //2

int index2 = message1.indexOf( “Hello”);
System.out.println(index2); // -1

int index3 = message2.indexOf(“Hello”);
System.out.println(index3); //0
```



String Equality

```
String s1 = "Hello";
String s2 = "Bye";
String s3 = s2;
String s4 = "Bye";
String s5 = new String("Bye");
```

- Use **equals()** to test if two strings have **the same characters in the same order**.
- Only use **==** to test if two strings refer to **the same object**.



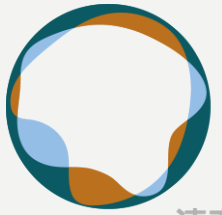
REVIEW -STRING

Method	Use
int length()	
String substring(int index1, int index2)	
String substring(int index2)	
int indexOf(string str1)	
int compareTo(String str2)	
boolean equals(String str2)	

EXAMPLE

```
String message1 = "This is a test";
String message2 = "Hello Class";

int len1 = message1.length();
System.out.println(len1);
int len2 = message2.length();
System.out.println(len2);
String sub1 =
message1.substring(0,3);
System.out.println(sub1);
String sub2 =
message1.substring(5);
System.out.println(sub2);
```

```
int index1 = message1.indexOf("is”);
System.out.println(index1);

int index2 =
message1.indexOf( “Hello”);
System.out.println(index2);

int index3 =
message2.indexOf(“Hello”);
System.out.println(index3);
```



WRAPPER CLASS

WRAPPER CLASS

- We can convert primitive types to object types using **wrapper classes**
- **A wrapper class** in JAVA is a class which contains or "wraps" primitive data types as an object.

Primitive type	Corresponding wrapper class
boolean	Boolean
char	Character
int	Integer
double	Double

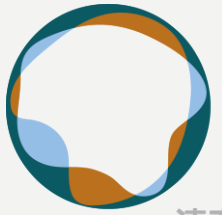
WRAPPER CLASS

Each wrapper class contains special value (like the minimum and maximum value for the type) and methods that are useful for converting between types.

For example:

Integer.MIN_VALUE

Integer.MAX_VALUE



WRAPPER CLASS

Create an object of wrapper class

```
Integer x = new Integer(7);
Double y = new Double(3.14);
```

Method:

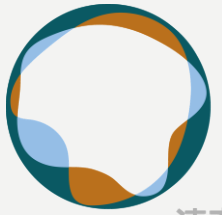
Integer.intValue()	Convert integer value into an int value
Double.doubleValue()	Convert Double value into a double value

WRAPPER CLASS

This conversion can actually be done AUTOMATICALLY at run time by the compiler using features called **autoboxing and unboxing**

Autoboxing: Converting a primitive value into an object of the corresponding wrapper class.

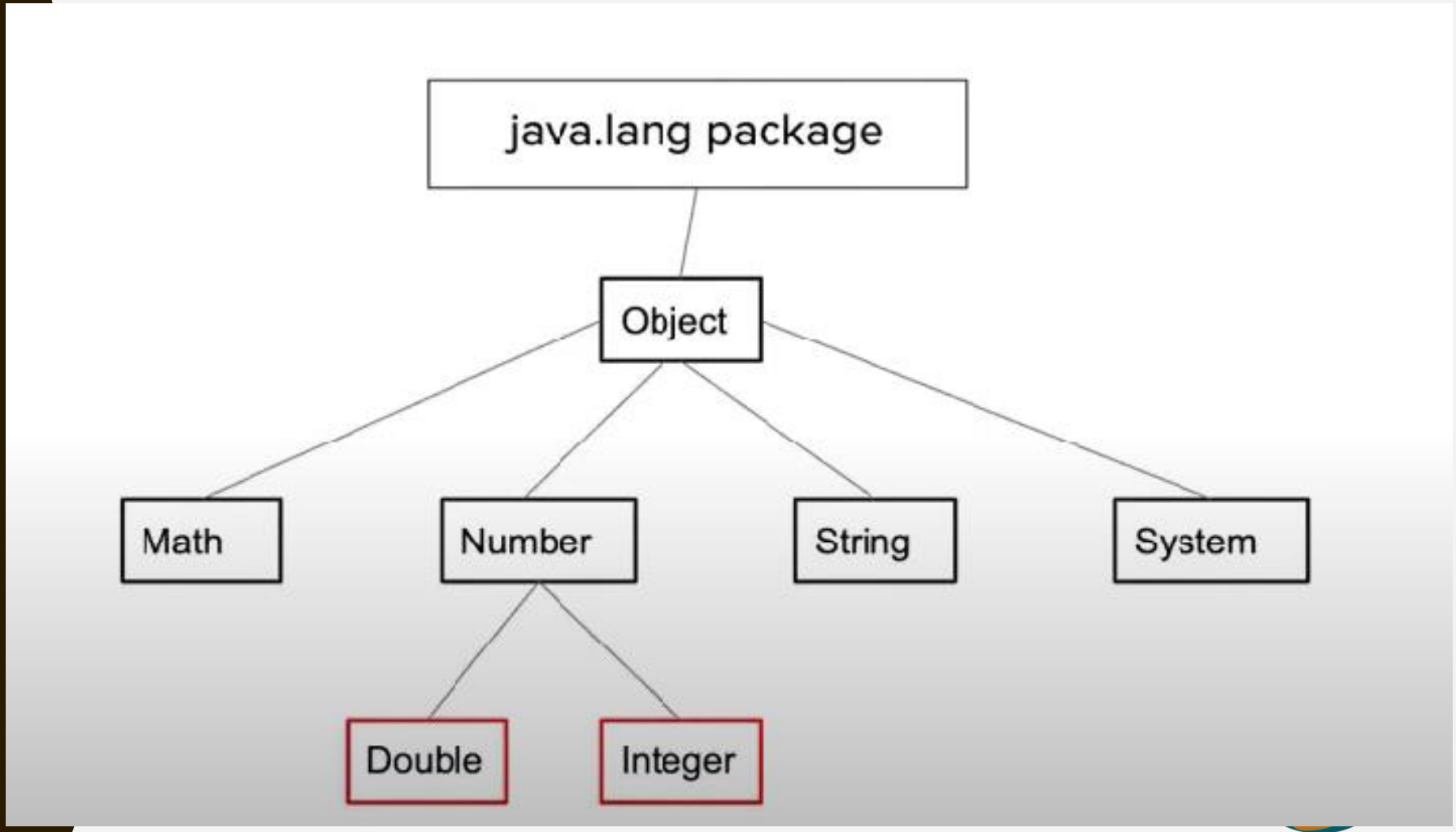
Unboxing: Converting an object of the wrapper type to its corresponding primitive value.



WRAPPER

```
Integer a = new Integer(5);
int x = a.intValue(); // unboxing x = 5
int y = a; // auto-unboxing, easier.

Integer b = new Integer(7); // boxing
Integer c = 7; // auto-boxing
int z = a + x; // auto-unboxing

Double d = new Double(7.5); // boxing
double e = d.doubleValue(); // unboxing
double f = d + 2.0; // auto-unboxing
```

JAVA.LANG. PACKAGE



CLASS-WORK

CodeHS:

2.9.6

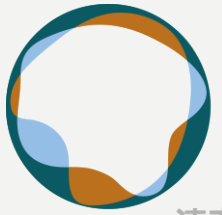
2.9.7

2.9.8

2.10.6

2.10.7

2.10.8



清森学校
BEIJING QINGSEN SCHOOL