AP CSA

zhang si 张思

zhangsi@rdfz.cn

ICC 609

4-standard classes

- Class libraries & package (Optional)
    - important declaration
    - input(scanner)/output(format output)
- String class method
- Wrapper class method
    - Integer class method
    - Double class method
- Object class method
    - toString()
    - equals()
- Math class method
    - Math.random()

Class Libraries

- A *class library* is a collection of classes that we can use when developing programs
- The *Java standard class library* is part of any Java development environment
- Its classes are not part of the Java language *perse*, but we rely on them heavily
- The `System` class and the `String` class are part of the Java standard class library
- Other class libraries can be obtained through third party vendors, or you can create them yourself

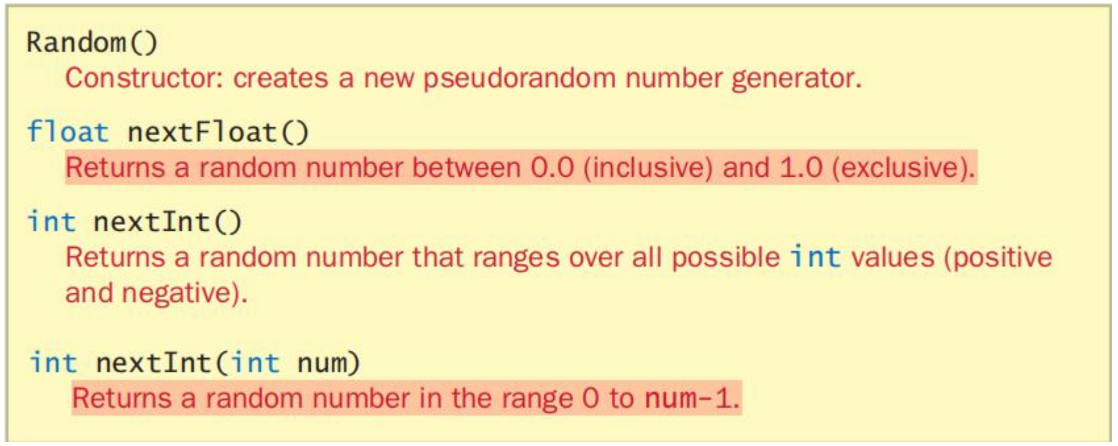
Packages

- The classes of the Java standard class library are organized into packages
- Some of the packages in the standard class library are:

Package	Purpose
java.lang	General support
java.applet	Creating applets for the web
java.awt	Graphics and graphical user interfaces
javax.swing	Additional graphics capabilities and components
java.net	Network communication
java.util	Utilities
javax.xml.parsers	XML document processing

The important Declaration

- All classes of the `java.lang` package are imported automatically into all programs
- That's why we didn't have to import the `System`, `String`, `Math`, `Integer` or `Double` classes explicitly in earlier programs
- The `Random` class is part of the `java.util` package
- It provides methods that generate pseudorandom numbers



```
Random()
```
Constructor: creates a new pseudorandom number generator.

```
float nextFloat()
```
Returns a random number between 0.0 (inclusive) and 1.0 (exclusive).

```
int nextInt()
```
Returns a random number that ranges over all possible `int` values (positive and negative).

```
int nextInt(int num)
```
Returns a random number in the range 0 to `num-1`.

The important Declaration

- When you want to use a class from a package, you could use its *fully qualified name*

```
java.util.Random
```

- Or you can *import* the class, and then use just the class name

```
import java.util.Random;
```

- To import all classes in a particular package, you can use the * wildcard character

```
import java.util.*;
```

- the general declaration for package and class:

```
import packageName.className;

import packageName.*;
```

Interactive Programs

- The `Scanner` class is used to get input from the user, allowing a program to be interactive
- It is part of the `java.util` package
- First a `Scanner` object is created

```
Scanner scan = new Scanner (System.in);
```

- Then various methods can be used to read different types of data from the keyboard

```
int num = scan.nextInt();
```

- See Quadratic.java (page 145)

Formatting Output

- The `NumberFormat` class has static methods that return a formatter object

  `getCurrencyInstance()`

  `getPercentInstance()`

- Each formatter object has a method called `format` that returns a string with the specified information in the appropriate format
- See Purchase.java (page 159)

```
import java.text.NumberFormat;
final double TAX_RATE = 0.06;
final double TAX = 19.8
NumberFormat fmt1 = NumberFormat.getCurrencyInstance();
 NumberFormat fmt2 = NumberFormat.getPercentInstance();
System.out.println(fmt2.format(TAX_RATE));//result is 6%
System.out.println(fmt1.format(TAX));//result is ￥19.8
```

Formatting Output

- `System.out.printf()` method allows the user to print a formatted string containing data values.

-
  ```
  System.out.printf("ID: %5d\tName: %s", id, name);
  ```

format string

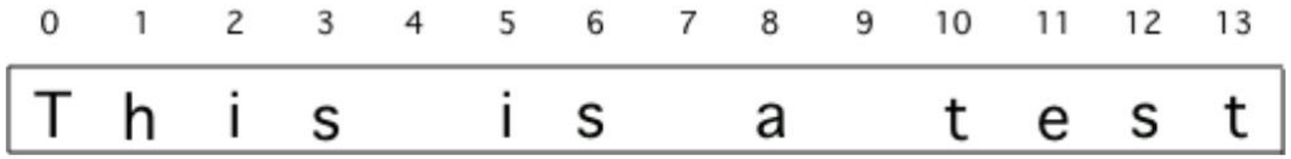
remaining parameters specify the values that are inserted into the format string

  - The first parameter specifies the format of the output and includes literal characters that label the output values as well as escape characters such as \t.
  - The pattern %5d indicates that the corresponding numeric value (id) should be printed in a field of five characters. if the numeric value is float, we can use %f (%.2f means rounds to two digits)
  - The pattern %s matches the string parameter name . The values of id and name are inserted into the string, producing a result such as:

    -
      ```
      ID: 24036 Name: Larry Flagelhopper
      ```

String Class

- A string holds characters in a sequence.
- Each character is at a position or **index** which **starts with 0** as shown below.



Note

The first character in a string is at index 0 and the last characters is at the length - 1.

- construct a new string object

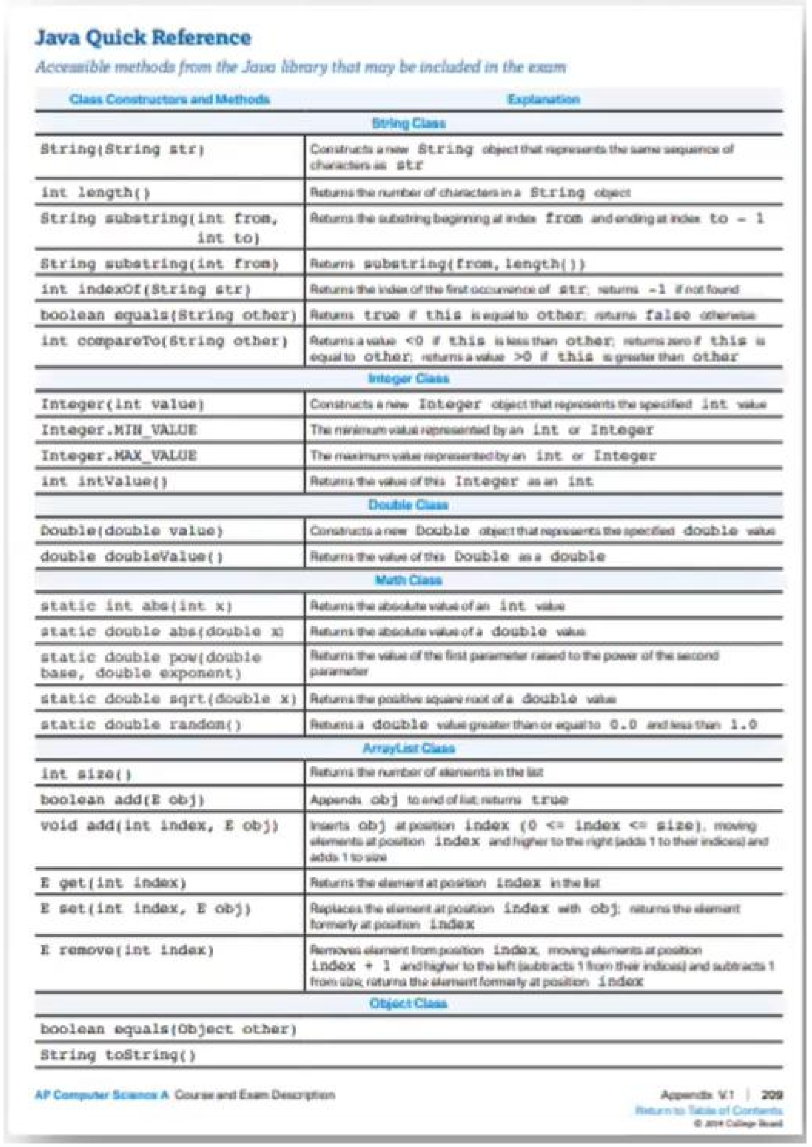
```
String title = new String ("Java Software Solutions");
```

- Because strings are so common, we don't have to use the `new` operator to create a `String` object

```
String title = "Java Software Solutions";
```

- This is special syntax that works only for strings

The AP CSA Java Quick Reference (JQR)



Java Quick Reference

Accessible methods from the Java library that may be included in the exam

Class Constructors and Methods	Explanation
String Class	
`String(String str)`	Constructs a new `String` object that represents the same sequence of characters as `str`
`int length()`	Returns the number of characters in a `String` object
`String substring(int from, int to)`	Returns the substring beginning at index `from` and ending at index `to - 1`
`String substring(int from)`	Returns `substring(from, length())`
`int indexOf(String str)`	Returns the index of the first occurrence of `str`; returns `-1` if not found
`boolean equals(String other)`	Returns `true` if `this` is equal to `other`; returns `false` otherwise
`int compareTo(String other)`	Returns a value `<0` if `this` is less than `other`; returns zero if `this` is equal to `other`; returns a value `>0` if `this` is greater than `other`
Integer Class	
`Integer(int value)`	Constructs a new `Integer` object that represents the specified `int` value
`Integer.MIN_VALUE`	The minimum value represented by an `int` or `Integer`
`Integer.MAX_VALUE`	The maximum value represented by an `int` or `Integer`
`int intValue()`	Returns the value of this `Integer` as an `int`
Double Class	
`Double(double value)`	Constructs a new `Double` object that represents the specified `double` value
`double doubleValue()`	Returns the value of this `Double` as a `double`
Math Class	
`static int abs(int x)`	Returns the absolute value of an `int` value
`static double abs(double x)`	Returns the absolute value of a `double` value
`static double pow(double base, double exponent)`	Returns the value of the first parameter raised to the power of the second parameter
`static double sqrt(double x)`	Returns the positive square root of a `double` value
`static double random()`	Returns a `double` value greater than or equal to `0.0` and less than `1.0`
ArrayList Class	
`int size()`	Returns the number of elements in the list
`boolean add(E obj)`	Appends `obj` to end of list; returns `true`
`void add(int index, E obj)`	Inserts `obj` at position `index` `(0 <= index <= size)`, moving elements at position `index` and higher to the right (adds 1 to their indices) and adds 1 to size
`E get(int index)`	Returns the element at position `index` in the list
`E set(int index, E obj)`	Replaces the element at position `index` with `obj`; returns the element formerly at position `index`
`E remove(int index)`	Removes element from position `index`, moving elements at position `index + 1` and higher to the left (subtracts 1 from their indices) and subtracts 1 from size; returns the element formerly at position `index`
Object Class	
`boolean equals(Object other)`	
`String toString()`	

AP Computer Science A Course and Exam Description

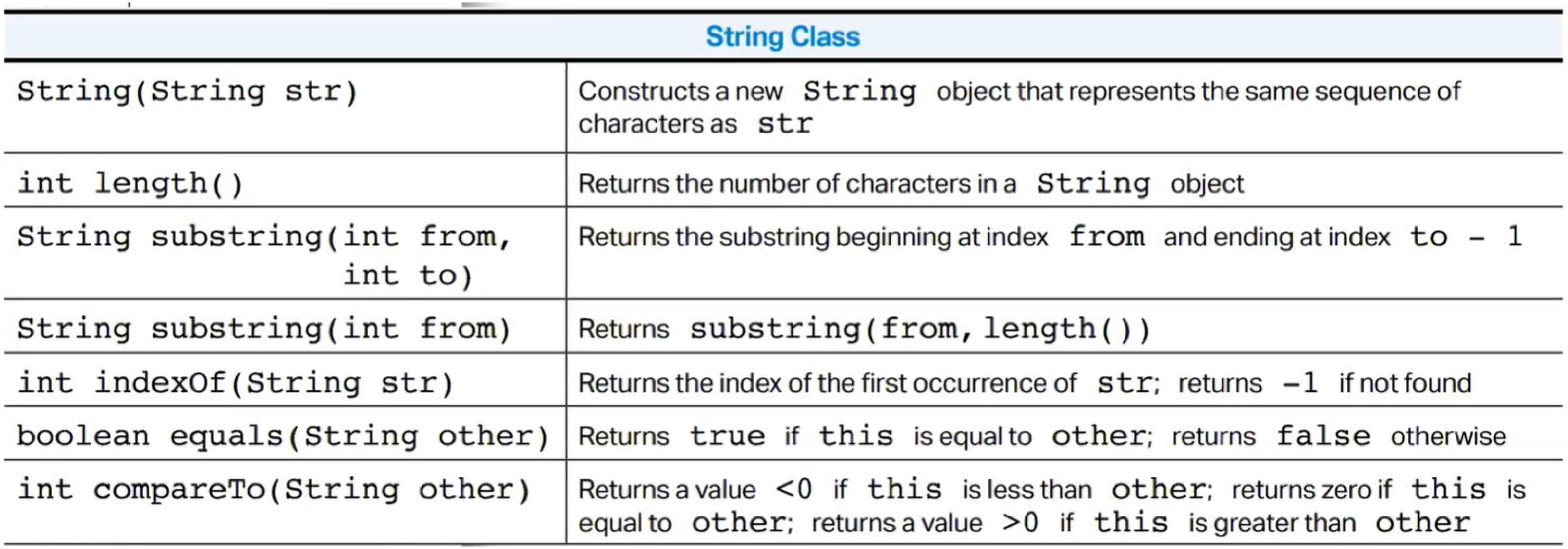
Appendix V.1 | 209

Return to Table of Contents

© 2019 College Board

- is a one-page document provided in the Course and Exam Description (page 209)
- is downloadable from AP Central's CSA course home
- lists each Class and method required in the AP CSA "subset" of Java and provides and example of the syntax and a description for each method
    - ✓ The CSA Java Subset is the listing of testable content for the CSA course.
- serves as our CSA version of an application programming interface

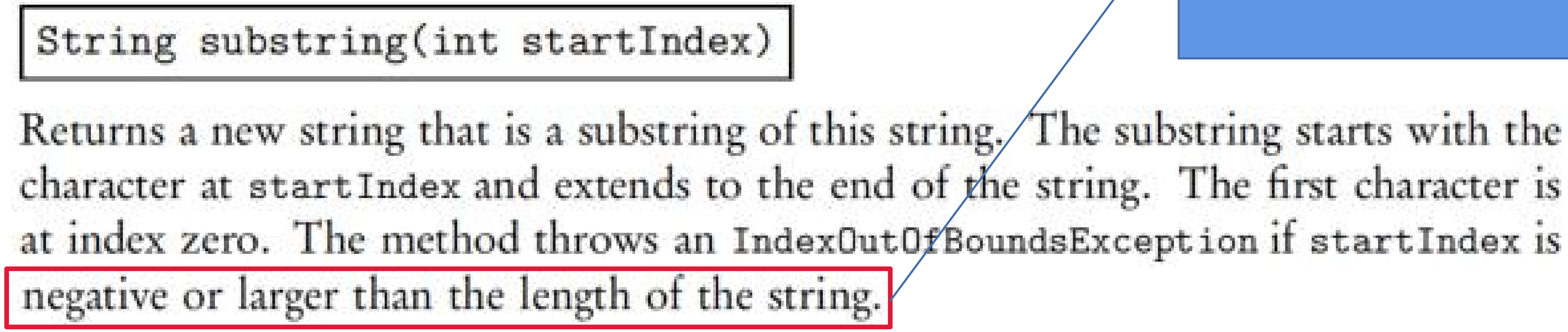
JQR String class



String Class	
`String(String str)`	Constructs a new `String` object that represents the same sequence of characters as `str`
`int length()`	Returns the number of characters in a `String` object
`String substring(int from, int to)`	Returns the substring beginning at index `from` and ending at index `to - 1`
`String substring(int from)`	Returns `substring(from, length())`
`int indexOf(String str)`	Returns the index of the first occurrence of `str`; returns `-1` if not found
`boolean equals(String other)`	Returns `true` if `this` is equal to `other`; returns `false` otherwise
`int compareTo(String other)`	Returns a value `<0` if `this` is less than `other`; returns zero if `this` is equal to `other`; returns a value `>0` if `this` is greater than `other`

String method

which means the only one parameters in substring method can be str.length().



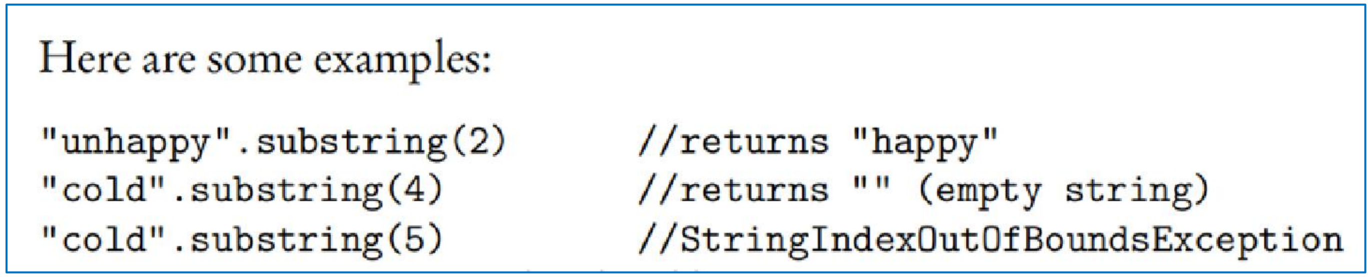
```
String substring(int startIndex)
```

Returns a new string that is a substring of this string. The substring starts with the character at `startIndex` and extends to the end of the string. The first character is at index zero. The method throws an `IndexOutOfBoundsException` if `startIndex` is negative or larger than the length of the string.

examples:

The result is: “”(empty string)

```
String str="1234567890";
String str1=str.substring(str.length());
System.out.println(str1);
```

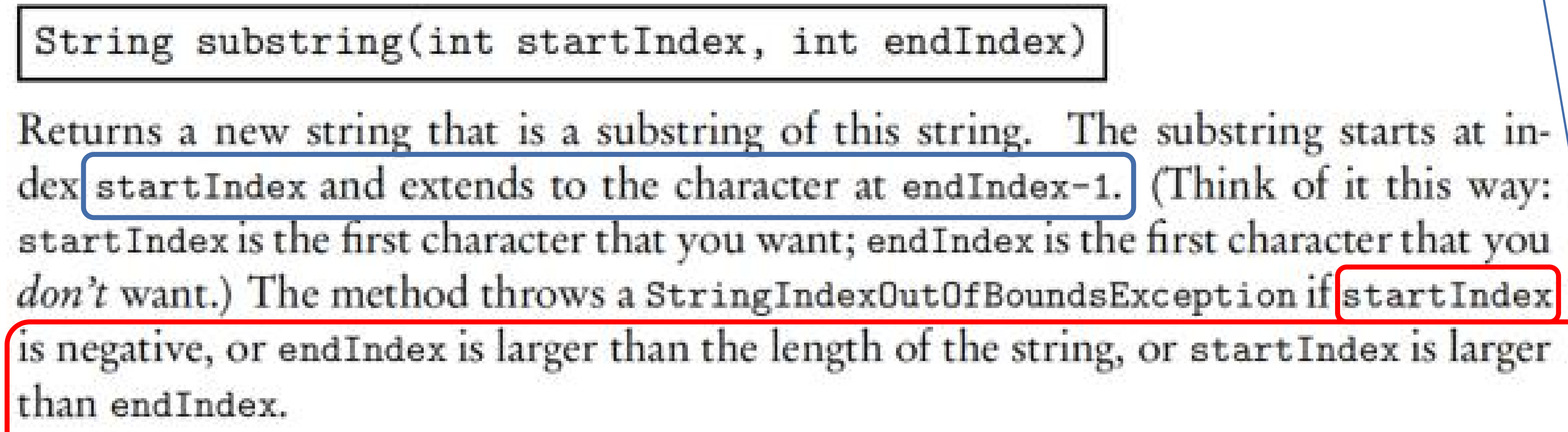


Here are some examples:

```
"unhappy".substring(2)      //returns "happy"
"cold".substring(4)         //returns "" (empty string)
"cold".substring(5)         //StringIndexOutOfBoundsException
```

String method

which means the following condition can compile:
starindex = endindex=string.length()



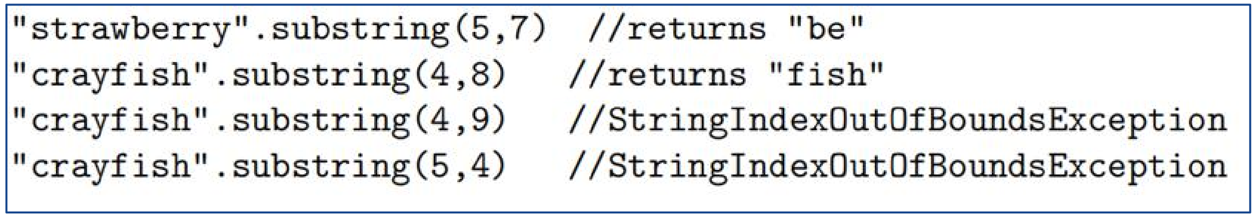
```
String substring(int startIndex, int endIndex)
```

Returns a new string that is a substring of this string. The substring starts at index `startIndex` and extends to the character at `endIndex-1`. (Think of it this way: `startIndex` is the first character that you want; `endIndex` is the first character that you *don't* want.) The method throws a `StringIndexOutOfBoundsException` if `startIndex` is negative, or `endIndex` is larger than the length of the string, or `startIndex` is larger than `endIndex`.

examples:

```
String str="1234567890";
String str1=str.substring(str.length(),str.length());
System.out.println(str1);
```

The result is: “”(empty string)



```
"strawberry".substring(5,7)  //returns "be"
"crayfish".substring(4,8)   //returns "fish"
"crayfish".substring(4,9)   //StringIndexOutOfBoundsException
"crayfish".substring(5,4)   //StringIndexOutOfBoundsException
```

String method

`int length()`

Returns the length of this string.

`int indexOf(String str)`

Returns the index of the first occurrence of `str` within this string. If `str` is not a substring of this string, `-1` is returned. The method throws a `NullPointerException` if `str` is null.

examples:

```
String s = "funnyfarm";
int x = s.indexOf("farm");  //x has value 5
x = s.indexOf("farmer");    //x has value -1
int y = s.length();         //y has value 9
```

String method

There are two ways to compare string objects:

boolean equals (String str)

returns true if this string contains exactly the same characters in same order as str ,and false otherwise

int compareTo (String other)

returns a value <0 if **`this`** is less than **`other`**,(means **`this`** is lexically before **`other`**);
returns 0 if **`this`** is equal to **`other`**;
returns a value>0 if **`this`** is greater than **`other`**(means **`this`** is lexically after **`other`**).

Characters are compared according to their position in the Unicode character set. All you need to know is that :

all digits precede all capital letters, which precede all lowercase letters.

Thus "5" comes before "R", which comes before "a".

Lexicographic Ordering

- Because comparing characters and strings is based on a character set, it is called a *lexicographic ordering*
- This is not strictly alphabetical when uppercase and lowercase characters are mixed
- For example, the string `"Great"` comes before the string `"fantastic"` because all of the uppercase letters come before all of the lowercase letters in Unicode

for Characters, All you need to know is that :
all digits precede all capital letters, which precede all lowercase letters.
Thus "5" comes before "R", which comes before "a".

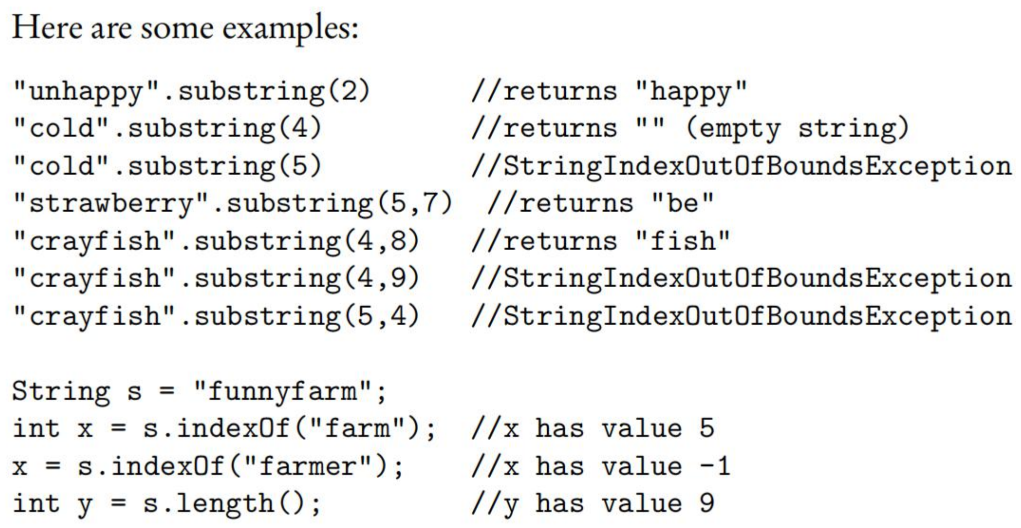
- Also, short strings come before longer strings with the same prefix (lexicographically)
- Therefore `"book"` comes before `"bookcase"`

examples:

```
String s1 = "HOT", s2 = "HOTEL", s3 = "dog";
if (s1.compareTo(s2) < 0))      //true, s1 terminates first
    ...
if (s1.compareTo(s3) > 0))      //false, "H" comes before "d"
```

String Method

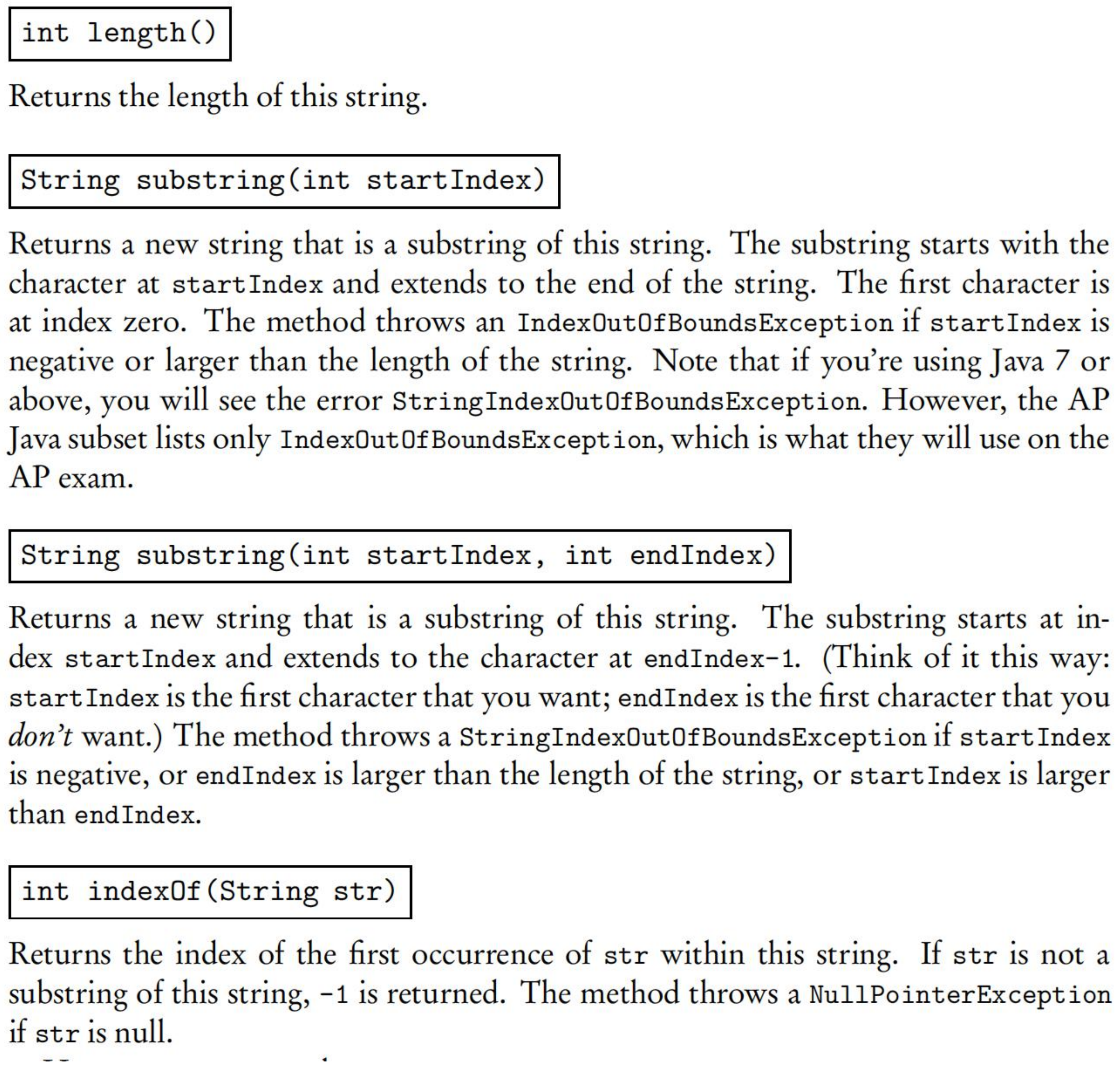
- Many of the methods *return a value*, such as an integer or a new `String` object
- All of the following are included in the quick reference that you get during the exam
- but you need to be familiar with them



Here are some examples:

```
"unhappy".substring(2)      //returns "happy"
"cold".substring(4)         //returns "" (empty string)
"cold".substring(5)         //StringIndexOutOfBoundsException
"strawberry".substring(5,7)  //returns "be"
"crayfish".substring(4,8)   //returns "fish"
"crayfish".substring(4,9)   //StringIndexOutOfBoundsException
"crayfish".substring(5,4)   //StringIndexOutOfBoundsException

String s = "funnyfarm";
int x = s.indexOf("farm");  //x has value 5
x = s.indexOf("farmer");    //x has value -1
int y = s.length();         //y has value 9
```



`int length()`

Returns the length of this string.

`String substring(int startIndex)`

Returns a new string that is a substring of this string. The substring starts with the character at `startIndex` and extends to the end of the string. The first character is at index zero. The method throws an `IndexOutOfBoundsException` if `startIndex` is negative or larger than the length of the string. Note that if you're using Java 7 or above, you will see the error `StringIndexOutOfBoundsException`. However, the AP Java subset lists only `IndexOutOfBoundsException`, which is what they will use on the AP exam.

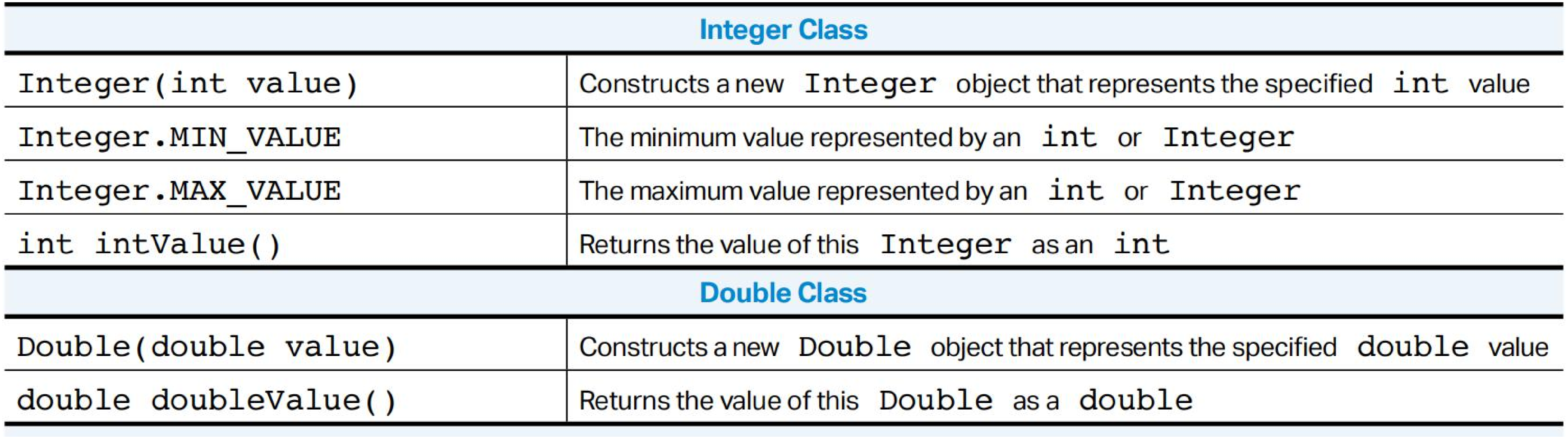
`String substring(int startIndex, int endIndex)`

Returns a new string that is a substring of this string. The substring starts at index `startIndex` and extends to the character at `endIndex-1`. (Think of it this way: `startIndex` is the first character that you want; `endIndex` is the first character that you *don't* want.) The method throws a `StringIndexOutOfBoundsException` if `startIndex` is negative, or `endIndex` is larger than the length of the string, or `startIndex` is larger than `endIndex`.

`int indexOf(String str)`

Returns the index of the first occurrence of `str` within this string. If `str` is not a substring of this string, `-1` is returned. The method throws a `NullPointerException` if `str` is null.

JQR Wrapper Class



Integer Class	
`Integer(int value)`	Constructs a new `Integer` object that represents the specified `int` value
`Integer.MIN_VALUE`	The minimum value represented by an `int` or `Integer`
`Integer.MAX_VALUE`	The maximum value represented by an `int` or `Integer`
`int intValue()`	Returns the value of this `Integer` as an `int`
Double Class	
`Double(double value)`	Constructs a new `Double` object that represents the specified `double` value
`double doubleValue()`	Returns the value of this `Double` as a `double`

Integer.MAX_VALUE = $2^{31}-1$
Integer.MIN_VALUE= -2^{32}

Wrapper class

- A *wrapper class* represents a particular primitive type
  - This is useful when a program requires an object instead of a primitive type (eg, **ArrayList**)
- For example

`Integer ageObj = new Integer (20);//`uses the `Integer` class to create an object which represents the integer 20 as an object

`int a=intValue(ageObj);//`convert the `Integer` object to int value(primitive type)

`Double ageObj1 = new Double(20.5);//`uses the `Double` class to create an object which effectively represents the double 20.5 as an object

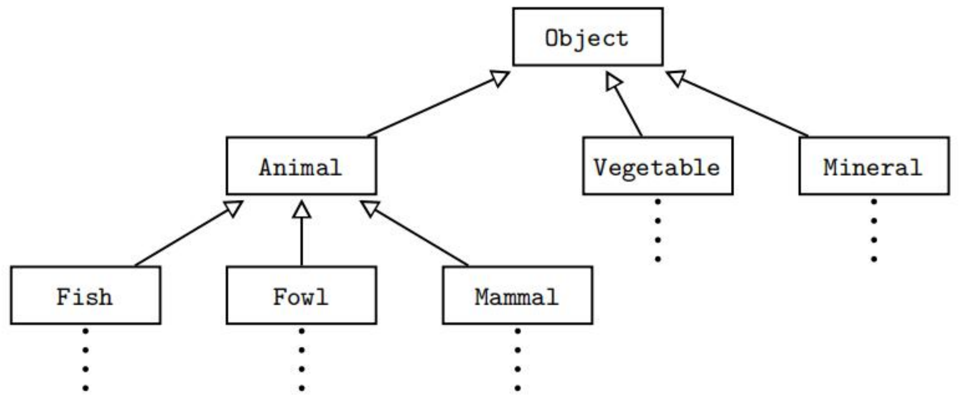
`double b=doubleValue(ageObj1);//`convert the `Double` object to double value(primitive type)

- ***Autoboxing*** automatically converts between wrapper classes and primitive types, so that the **following is also valid**:

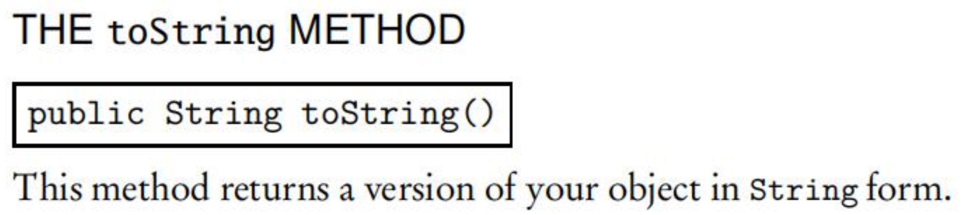
```
Integer ageObj = 20;
```

the Object class

- The Universal Superclass
- Think of `Object` class as the superclass of the universe. **Every class automatically extends `Object` class**, which means that `Object` class is a direct or indirect superclass of every other class.

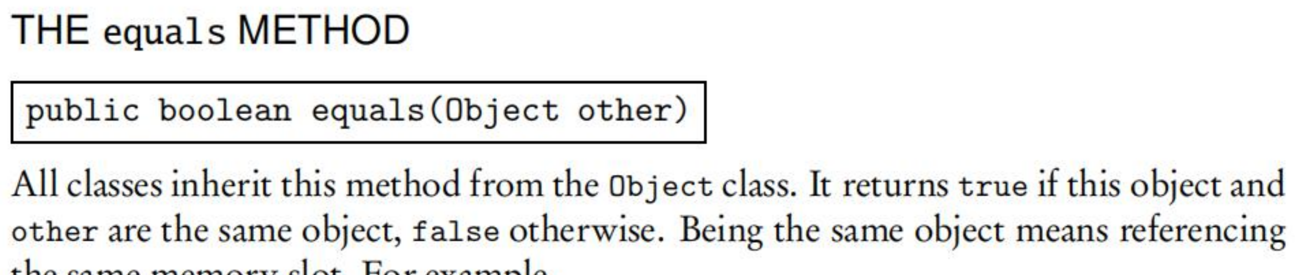


- method `in object` class


THE toString METHOD

```
public String toString()
```

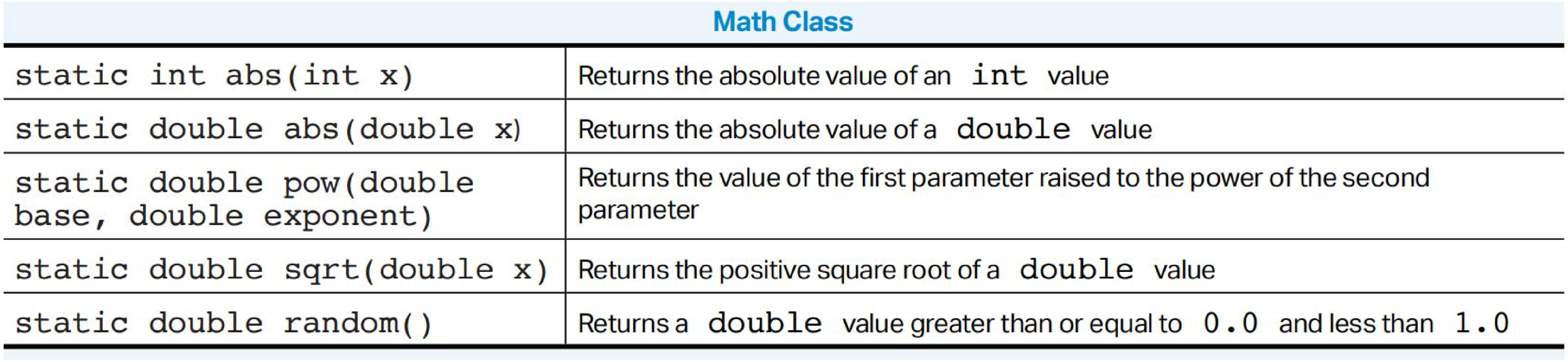
This method returns a version of your object in `String` form.


THE equals METHOD

```
public boolean equals(Object other)
```

All classes inherit this method from the `Object` class. It returns `true` if this object and `other` are the same object, `false` otherwise. Being the same object means referencing the same memory slot. For example,

JQR Math Class



Math Class	
`static int abs(int x)`	Returns the absolute value of an `int` value
`static double abs(double x)`	Returns the absolute value of a `double` value
`static double pow(double base, double exponent)`	Returns the value of the first parameter raised to the power of the second parameter
`static double sqrt(double x)`	Returns the positive square root of a `double` value
`static double random()`	Returns a `double` value greater than or equal to `0.0` and less than `1.0`

Math class

- Some methods can be invoked through the class name, instead of through an object of the class

- These methods are called *class methods* or *static methods*

- The `Math` class contains many static methods, providing various mathematical functions, such as absolute value, trigonometry functions, square root, etc.

```
temp = Math.cos(90) + Math.sqrt(delta);
```

- The Math class is part of the java.lang package.

Math class

`static int abs(int x)`

Returns the absolute value of integer x.

`static double abs(double x)`

Returns the absolute value of real number x.

`static double pow(double base, double exp)`

Returns $\texttt{base}^{\texttt{exp}}$. Assumes $\texttt{base} > 0$, or $\texttt{base} = 0$ and $\texttt{exp} > 0$, or $\texttt{base} < 0$ and `exp` is an integer.

`static double sqrt(double x)`

Returns $\sqrt{x}$, $x \geq 0$.

`static double random()`

Returns a random number r, where $0.0 \leq r < 1.0$.

calling method:
eg:

```
double a = 4.0;
System.out.println(Math.sqrt(a));
```

Math class

- Execise—Random Numbers
  - Example 1
  - Math.random() method can produce a random real value x in the range 0.0 ≤ x < 1.0.

- **Question： how to produce a random integer, from 5(inclusive) ato 24(exclusive).**
- （hits： Using a cast to int, a scaling factor, and a shifting value, Math.random() can be used to produce random integers in any range.)
-

```
(int)(Math.random()*20)+5
 (int)(Math.random()*20+5)
```