Java Package

Agenda:

Inner class, Dynamic Binding, Java Recursion,
Java Enum, Java Dates, Java Math class
Wrapper classes : Auto boxing and Unboxing
Casting–Up-casting and Down-Casting
The Object Class in Java, Object Cloning
javadoc tool - Creating API Document

Md. Mamun Hossain
B.Sc. (Engg.) & M.Sc. (Thesis) in CSE , SUST
Assistant Professor, Dept. of CSE, BAUST

If you import a package, sub-packages will not be imported.

Java OOP : Miscellaneous

Miscellaneous

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Java OOPs Miscellaneous

Assignment: Some of these concepts keeps as HW for the students

- **Inner class**
- Dynamic Binding
- Java Recursion
- Java Enum
- Java Dates
- Java Math class
- Wrapper classes – Auto boxing and Unboxing
- **Casting–Up-casting and Down-Casting**
- The Object Class in Java
- Object Cloning
- javadoc tool - Creating API Document

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

OOP : Miscellaneous

Java Inner class

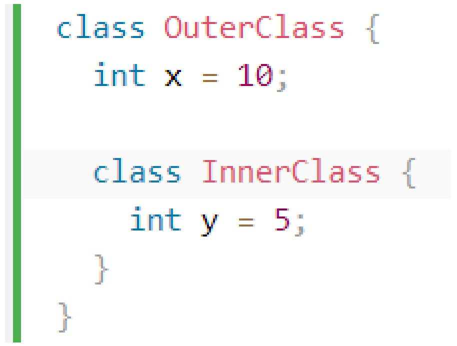
a class within a class

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

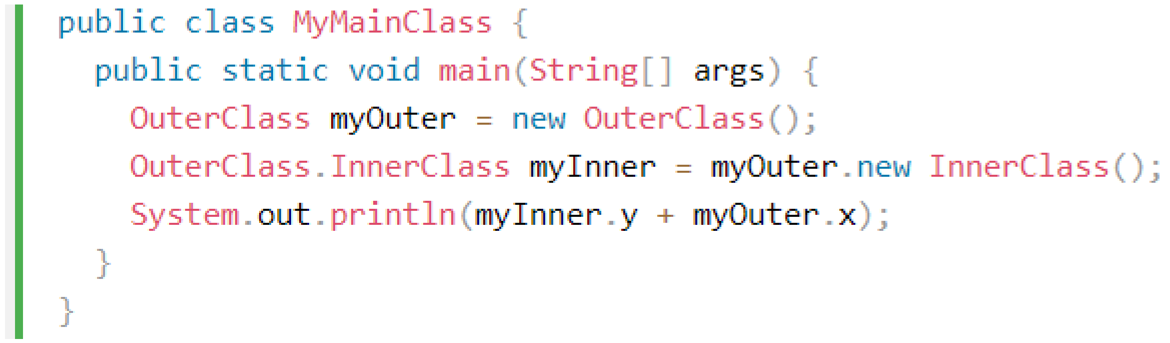
Java Inner class

In Java, it is also possible to nest classes (a class within a class). The purpose of nested classes is to group classes that belong together, which makes your code more readable and maintainable.

To access the inner class, create an object of the outer class, and then create an object of the inner class:



```java
class OuterClass {
  int x = 10;

  class InnerClass {
    int y = 5;
  }
}
```



```java
public class MyMainClass {
  public static void main(String[] args) {
    OuterClass myOuter = new OuterClass();
    OuterClass.InnerClass myInner = myOuter.new InnerClass();
    System.out.println(myInner.y + myOuter.x);
  }
}
```

```
// Outputs 15 (5 + 10)
```

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Java OOPs Miscellaneous

Assignment: Some of these concepts keeps as HW for the students

- Inner class
- **Dynamic Binding**
- Java Recursion
- Java Enum
- Java Dates
- Java Math class
- Wrapper classes – Auto boxing and Unboxing
- **Casting–Up-casting and Down-Casting**
- The Object Class in Java
- Object Cloning
- javadoc tool - Creating API Document

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

OOP : Miscellaneous

Dynamic Binding

Connecting a method call to the method body

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Dynamic Binding

- **Binding**

Connecting a method call to the method body is known as binding.

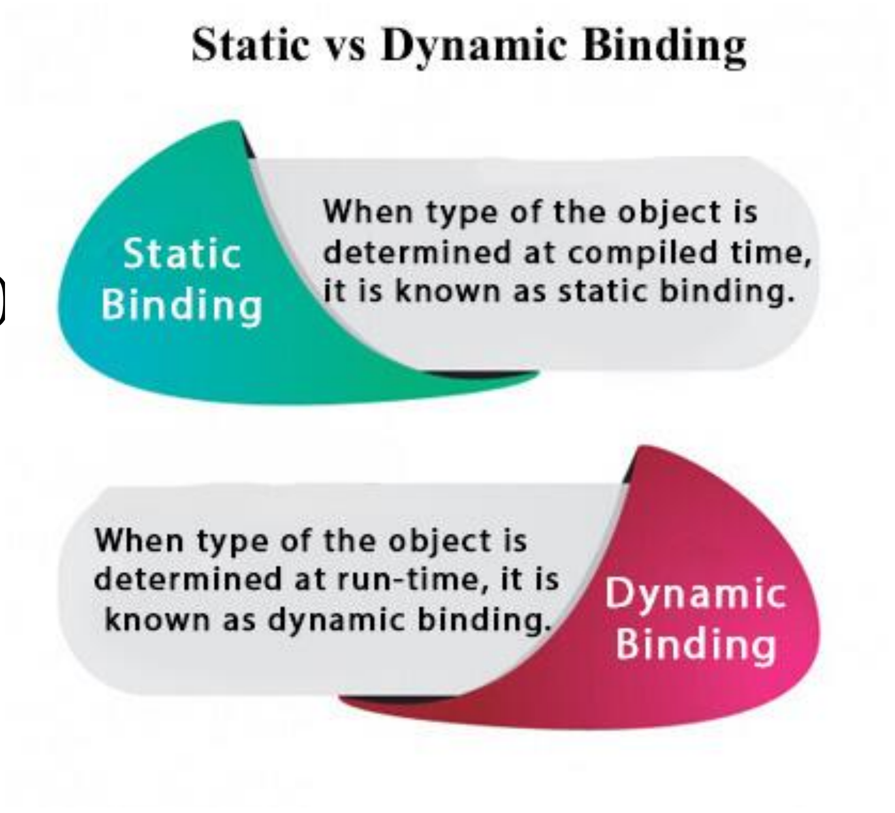
There are two types of binding

- **Static Binding**

(also known as Early Binding)

- **Dynamic Binding**

(also known as Late Binding).


Static vs Dynamic Binding
Static Binding
When type of the object is determined at compiled time, it is known as static binding.
Dynamic Binding
When type of the object is determined at run-time, it is known as dynamic binding.

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Dynamic Binding : Understanding Type

1. variables have a type

Each variable has a type, it may be primitive and non-primitive.

int data=30; //*Here data variable is a type of int.*

2. References have a type

```
class Dog{
    public static void main(String args[]){
            Dog d1;//Here d1 is a type of Dog }}
```

3. Objects have a type

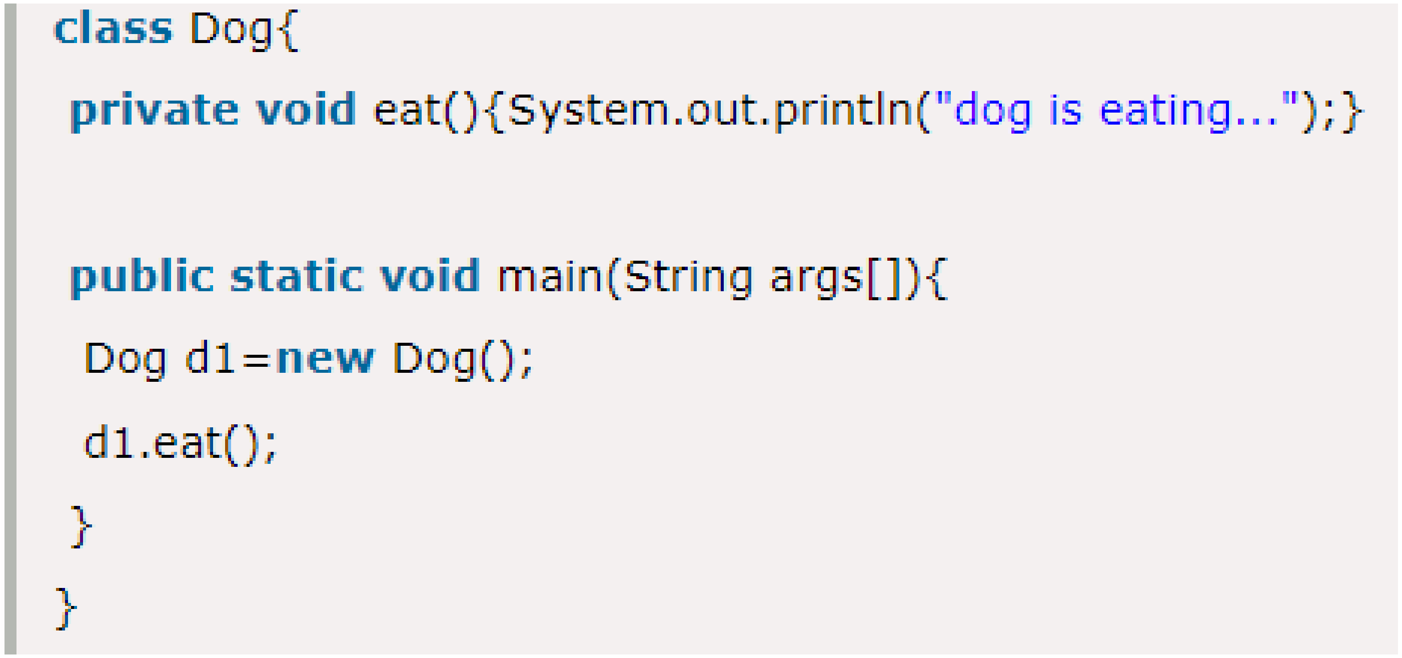
An object is an instance of particular java class, but it is also an instance of its superclass.

```
class Animal{}
class Dog extends Animal{
    public static void main(String args[]){
            Dog d1=new Dog();  }}
```

Here d1 is an instance of Dog class, but it is also an instance of Animal.

Static binding

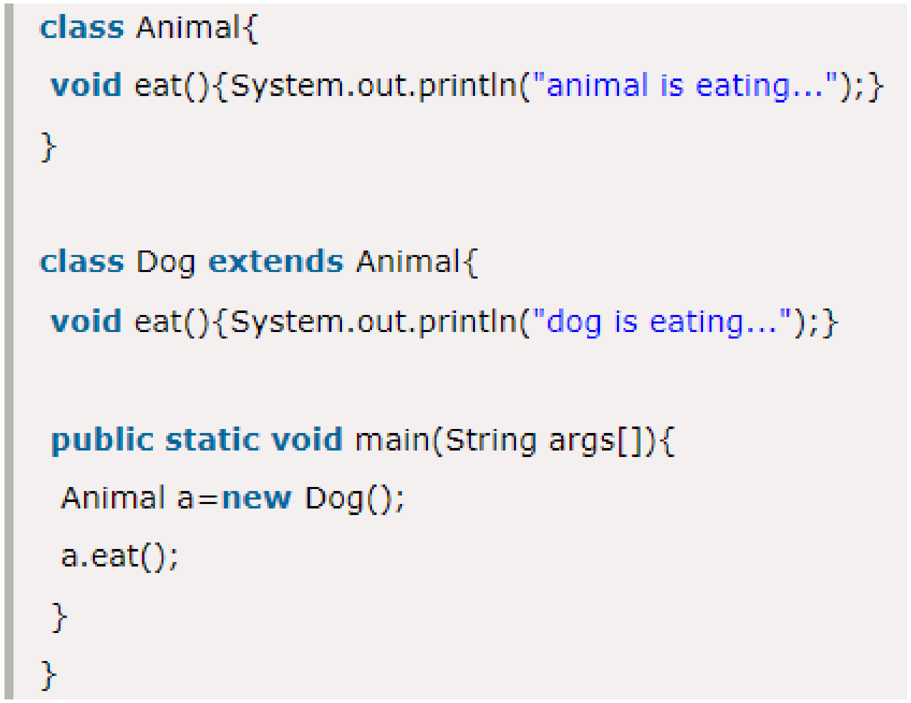
- When type of the object is determined at compiled time (by the compiler), it is known as static binding.
- If there is any ***private, final or static*** method in a class, there is static binding.



```
class Dog{
 private void eat(){System.out.println("dog is eating...");}

 public static void main(String args[]){
  Dog d1=new Dog();
  d1.eat();
 }
}
```

Dynamic binding

- When type of the object is determined at run-time, it is known as dynamic binding.



```
class Animal{
 void eat(){System.out.println("animal is eating...");}
}

class Dog extends Animal{
 void eat(){System.out.println("dog is eating...");}

 public static void main(String args[]){
  Animal a=new Dog();
  a.eat();
 }
}
```

In the above example object type cannot be determined by the compiler, because the instance of Dog is also an instance of Animal.

So compiler doesn't know its type, only its base type.

In the above example object type cannot be determined by the compiler, because the instance of Dog is also an instance of Animal. So compiler doesn't know its type, only its base type.

Java OOPs Miscellaneous

Assignment: Some of these concepts keeps as HW for the students

- Inner class
- Dynamic Binding
- **Java Recursion**
- Java Enum
- Java Dates
- Java Math class
- Wrapper classes – Auto boxing and Unboxing
- **Casting–Up-casting and Down-Casting**
- The Object Class in Java
- Object Cloning
- javadoc tool - Creating API Document

OOP : Miscellaneous

Java Recursion

function call itself

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Java Recursion

- ***Java Recursion***

Recursion is the technique of making a function call itself. This technique provides a way to break complicated problems down into simple problems which are easier to solve.

Recursion may be a bit difficult to understand. The best way to figure out how it works is to experiment with it.

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Java Recursion

Use recursion to add all of the numbers up to 10.

```
public class MyClass {
  public static void main(String[] args) {
    int result = sum(10);
    System.out.println(result);
  }
  public static int sum(int k) {
    if (k > 0) {
      return k + sum(k - 1);
    } else {
      return 0;
    }
  }
}
```

Example Explained

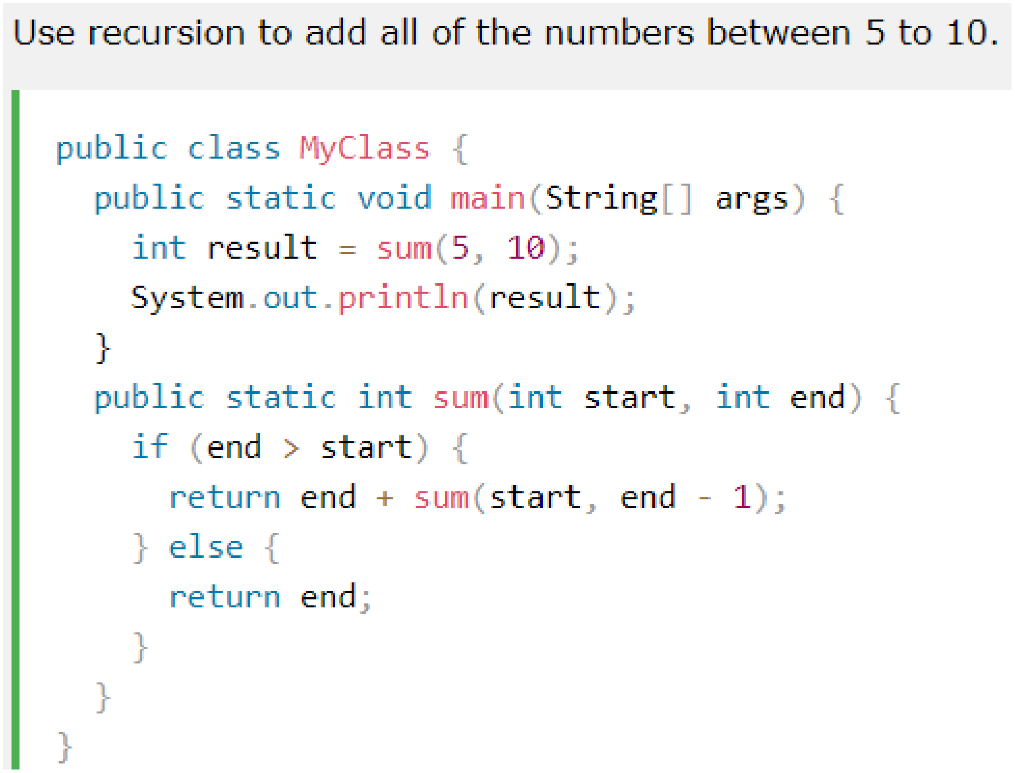
When the sum() function is called, it adds parameter k to the sum of all numbers smaller than k and returns the result. When k becomes 0, the function just returns 0. When running, the program follows these steps:

```
10 + sum(9)
10 + ( 9 + sum(8) )
10 + ( 9 + ( 8 + sum(7) ) )
...
10 + 9 + 8 + 7 + 6 + 5 + 4 + 3 + 2 + 1 + sum(0)
10 + 9 + 8 + 7 + 6 + 5 + 4 + 3 + 2 + 1 + 0
```

Since the function does not call itself when k is 0, the program stops there and returns the result.

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Java Recursion



Use recursion to add all of the numbers between 5 to 10.

```java
public class MyClass {
  public static void main(String[] args) {
    int result = sum(5, 10);
    System.out.println(result);
  }
  public static int sum(int start, int end) {
    if (end > start) {
      return end + sum(start, end - 1);
    } else {
      return end;
    }
  }
}
```

In this example, the function adds a range of numbers between a start and an end.
The halting condition for this recursive function is when **end** is not greater than **start**:

Halting Condition

Just as loops can run into the problem of infinite looping, recursive functions can run into the problem of infinite recursion. Infinite recursion is when the function never stops calling itself. Every recursive function should have a halting condition, which is the condition where the function stops calling itself. In the previous example, the halting condition is when the parameter k becomes 0.

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Java OOPs Miscellaneous

Assignment: Some of these concepts keeps as HW for the students

- Inner class
- Dynamic Binding
- Java Recursion
- **Java Enum**
- Java Dates
- Java Math class
- Wrapper classes – Auto boxing and Unboxing
- **Casting–Up-casting and Down-Casting**
- The Object Class in Java
- Object Cloning
- javadoc tool - Creating API Document

OOP : Miscellaneous

Java Enums

a special "class" that represents a group of **constant**

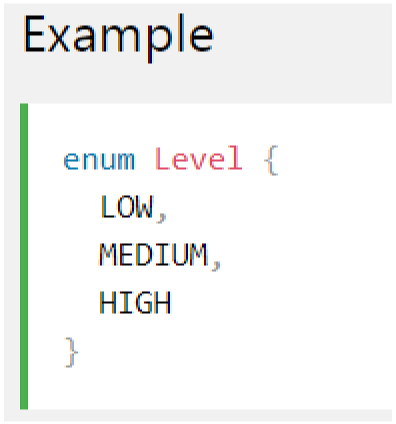
Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Java Enums

Enums

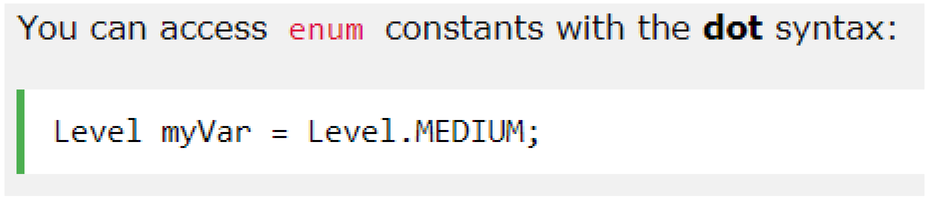
An enum is a special "class" that represents a group of **constants** (unchangeable variables, like final variables).

To create an enum, use the enum keyword (instead of class or interface), and separate the constants with a comma. Note that they should be in uppercase letters:



Example

```java
enum Level {
  LOW,
  MEDIUM,
  HIGH
}
```



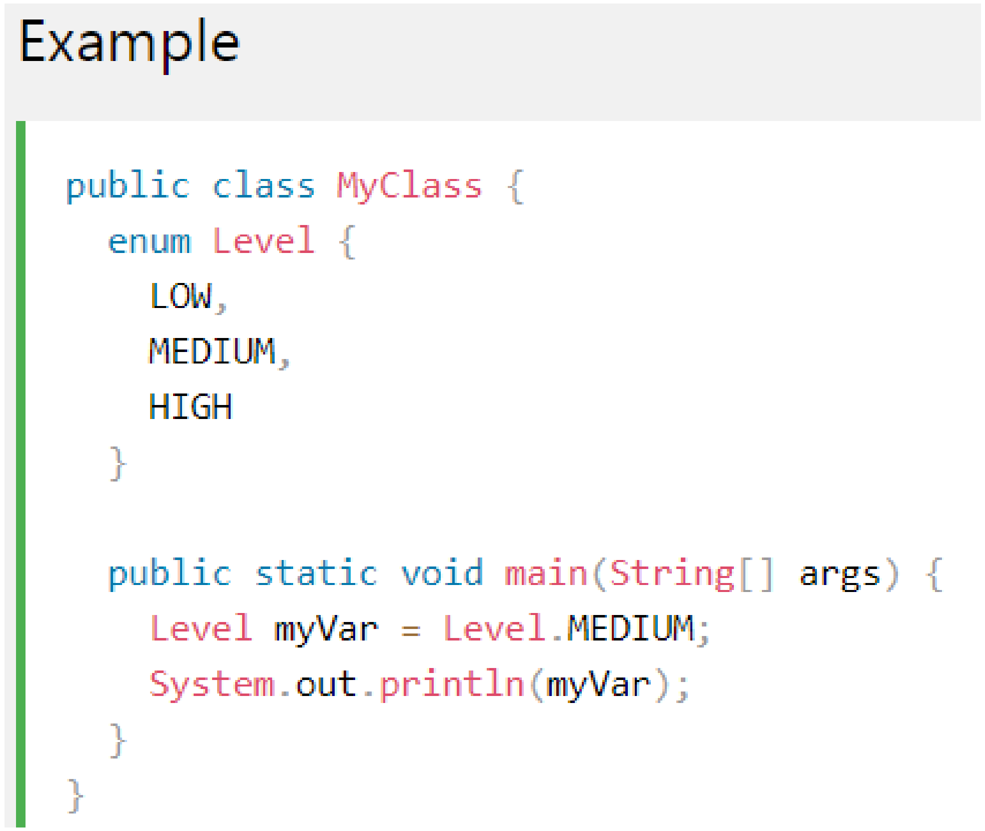
You can access `enum` constants with the **dot** syntax:

```java
Level myVar = Level.MEDIUM;
```

***Enum** is short for "enumerations", which means "specifically listed".*

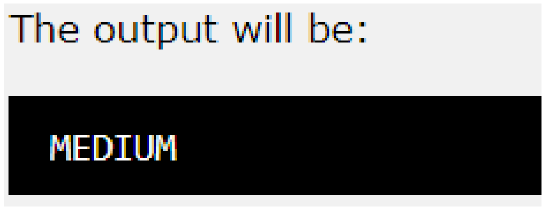
Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Java Enums : Example



Example

```
public class MyClass {
  enum Level {
    LOW,
    MEDIUM,
    HIGH
  }

  public static void main(String[] args) {
    Level myVar = Level.MEDIUM;
    System.out.println(myVar);
  }
}
```



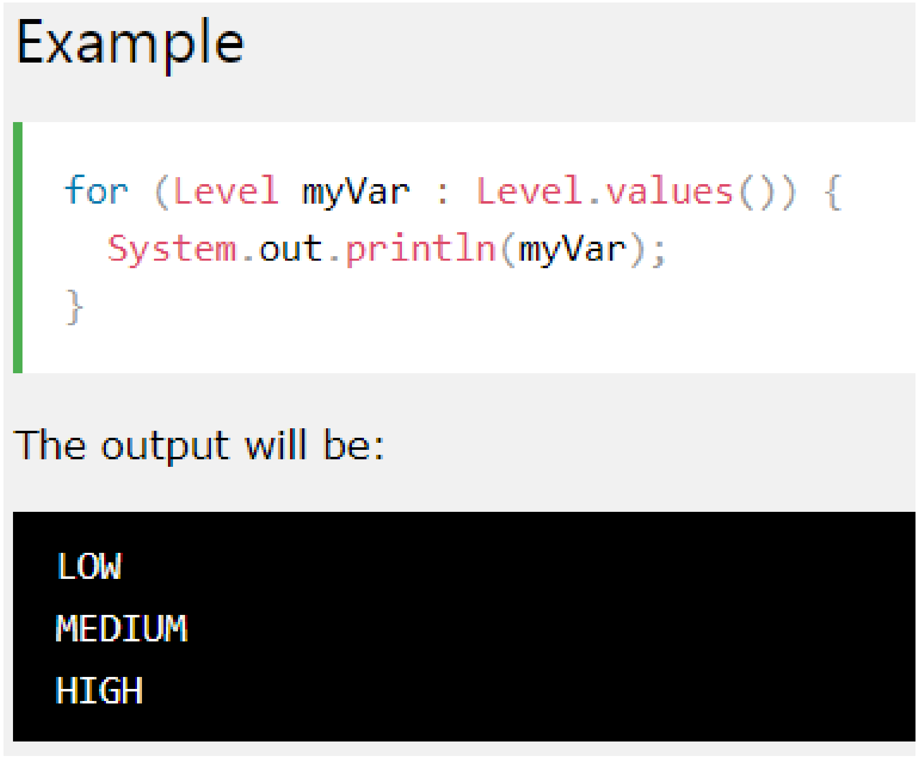
The output will be:

```
MEDIUM
```

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Loop Through an Enum

The enum type has a values() method, which returns an array of all enum constants. This method is useful when you want to loop through the constants of an enum:



Example

```
for (Level myVar : Level.values()) {
  System.out.println(myVar);
}
```

The output will be:

```
LOW
MEDIUM
HIGH
```

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Enum : Summary

- **Difference between Enums and Classes**
  - An enum can, just like a class, have attributes and methods. The only difference is that enum constants are public, static and final (unchangeable - cannot be overridden).
  - An enum cannot be used to create objects, and it cannot extend other classes (but it can implement interfaces).

- **Why And When To Use Enums?**

Use enums when you have values that you know aren't going to change, like month days, days, colors, deck of cards, etc.

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Java OOPs Miscellaneous

Assignment: Some of these concepts keeps as HW for the students

- Inner class
- Dynamic Binding
- Java Recursion
- Java Enum
- Java Dates
- **Java Math class**
- Wrapper classes – Auto boxing and Unboxing
- **Casting–Up-casting and Down-Casting**
- The Object Class in Java
- Object Cloning
- javadoc tool - Creating API Document

OOP : Miscellaneous

Java Math class

Java Math class provides several methods to work on math calculations like ***min(), max(), avg(), sin(), cos(), tan(), round(), ceil(), floor(), abs() etc.***

All Math methods are static.

Java Math class

The Java Math class has many methods that allows you to perform mathematical tasks on numbers.

Math.max(*x*,*y*)

The `Math.max(x,y)` method can be used to find the highest value of *x* and *y*:

Example

```
Math.max(5, 10);
```

Math.min(*x*,*y*)

The `Math.min(x,y)` method can be used to find the lowest value of of *x* and *y*:

Example

```
Math.min(5, 10);
```

Math.sqrt(*x*)

The `Math.sqrt(x)` method returns the square root of *x*:

Example

```
Math.sqrt(64);
```

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Java Math class

Math.abs(*x*)

The `Math.abs(x)` method returns the absolute (positive) value of *x*:

Example

```
Math.abs(-4.7);
```

Random Numbers

`Math.random()` returns a random number between 0.0 (inclusive), and 1.0 (exclusive):

Example

```
Math.random();
```

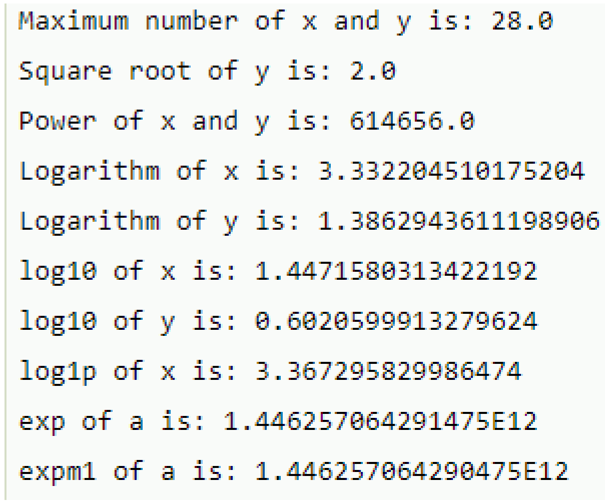
To get more control over the random number, e.g. you only want a random number between 0 and 100, you can use the following formula:

```
int randomNum = (int)(Math.random() * 101);  // 0 to 100
```

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Java Math class: Example

```
public class JavaMathExample1
{    public static void main(String[] args)
   {    double x = 28, y = 4;

        // return the maximum of two numbers
        System.out.println("Maximum number of x and y is: " +Math.max(x, y));

        // return the square root of y
        System.out.println("Square root of y is: " + Math.sqrt(y));

        //returns 28 power of 4 i.e. 28*28*28*28
        System.out.println("Power of x and y is: " + Math.pow(x, y));

        // return the logarithm of given value
        System.out.println("Logarithm of x is: " + Math.log(x));
        System.out.println("Logarithm of y is: " + Math.log(y));

  }
}
```

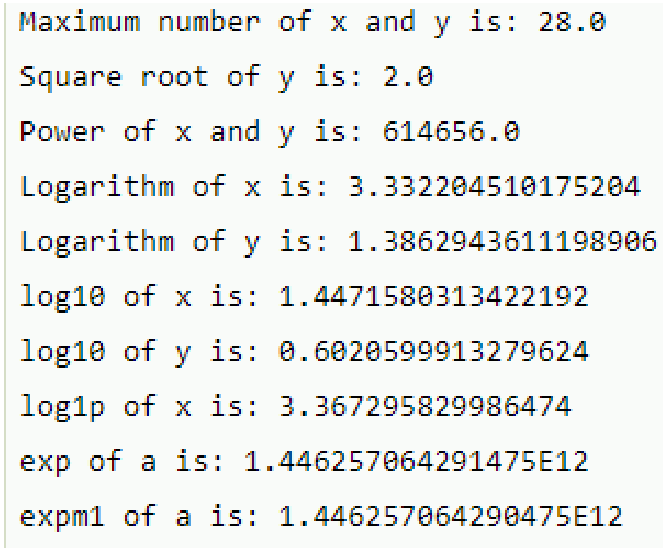


```
Maximum number of x and y is: 28.0
Square root of y is: 2.0
Power of x and y is: 614656.0
Logarithm of x is: 3.332204510175204
Logarithm of y is: 1.3862943611198906
log10 of x is: 1.4471580313422192
log10 of y is: 0.6020599913279624
log1p of x is: 3.367295829986474
exp of a is: 1.446257064291475E12
expm1 of a is: 1.446257064290475E12
```

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Java Math class: Example

```
public class JavaMathExample2
{    public static void main(String[] args)
   {    double x = 28, y = 4;

        // return the logarithm of given value when base is 10

        System.out.println("log10 of x is: " + Math.log10(x));
        System.out.println("log10 of y is: " + Math.log10(y));


        // return the log of x + 1
        System.out.println("log1p of x is: " +Math.log1p(x));

        // return a power of 2
        System.out.println("exp of a is: " +Math.exp(x));

        // return (a power of 2)-1
        System.out.println("expm1 of a is: " +Math.expm1(x));

   }
}
```

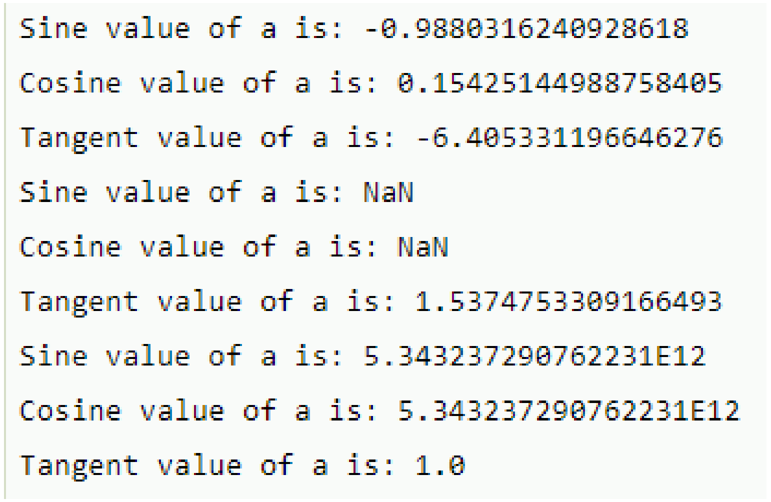


```
Maximum number of x and y is: 28.0

Square root of y is: 2.0

Power of x and y is: 614656.0

Logarithm of x is: 3.332204510175204

Logarithm of y is: 1.3862943611198906

log10 of x is: 1.4471580313422192

log10 of y is: 0.6020599913279624

log1p of x is: 3.367295829986474

exp of a is: 1.446257064291475E12

expm1 of a is: 1.446257064290475E12
```

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Java Math class: Example

```
double a = 30;
// converting values to radian
double b = Math.toRadians(a);
// return the trigonometric sine of a
System.out.println("Sine value of a is: " +Math.sin(a));
// return the trigonometric cosine value of a
System.out.println("Cosine value of a is: " +Math.cos(a));
// return the trigonometric tangent value of a
System.out.println("Tangent value of a is: " +Math.tan(a));
// return the trigonometric arc sine of a
System.out.println("Sine value of a is: " +Math.asin(a));
// return the trigonometric arc cosine value of a
System.out.println("Cosine value of a is: " +Math.acos(a));
// return the trigonometric arc tangent value of a
System.out.println("Tangent value of a is: " +Math.atan(a));
// return the hyperbolic sine of a
System.out.println("Sine value of a is: " +Math.sinh(a));
// return the hyperbolic cosine value of a
System.out.println("Cosine value of a is: " +Math.cosh(a));
// return the hyperbolic tangent value of a
System.out.println("Tangent value of a is: " +Math.tanh(a));
```



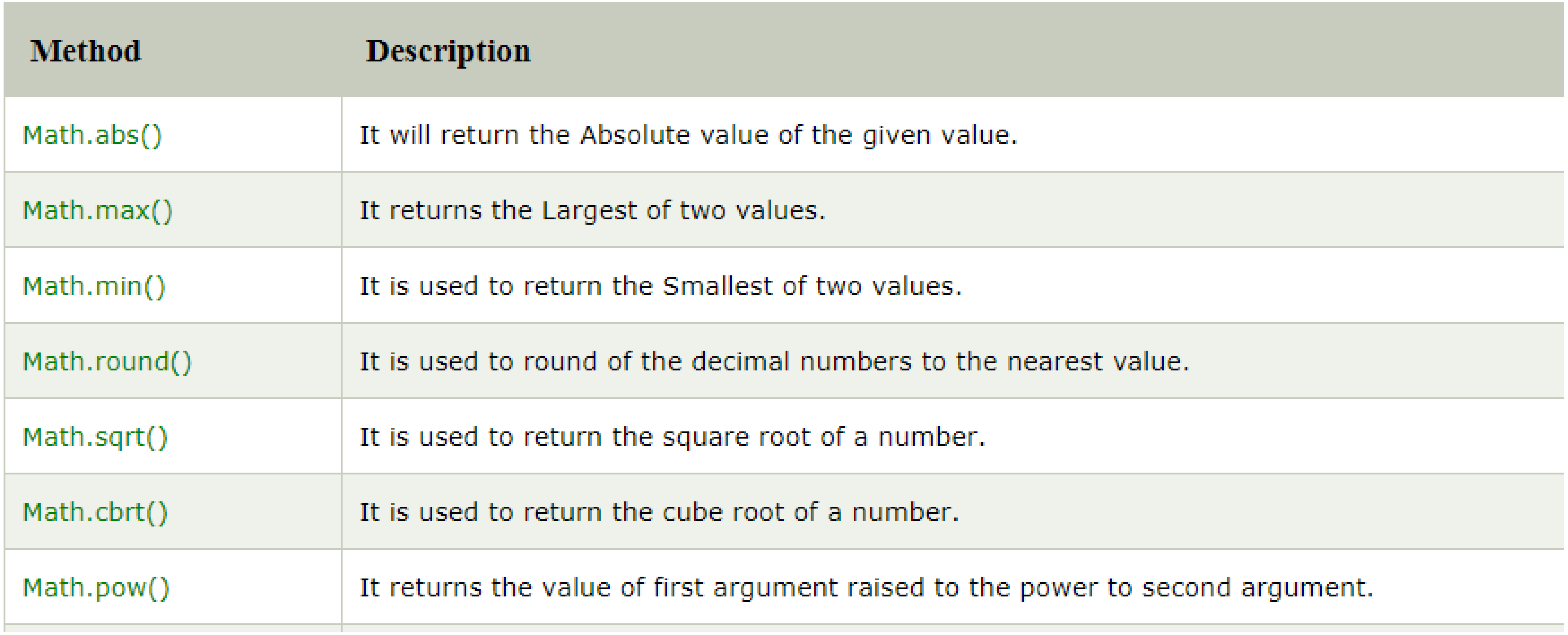
```
Sine value of a is: -0.9880316240928618

Cosine value of a is: 0.15425144988758405

Tangent value of a is: -6.405331196646276

Sine value of a is: NaN

Cosine value of a is: NaN

Tangent value of a is: 1.5374753309166493

Sine value of a is: 5.343237290762231E12

Cosine value of a is: 5.343237290762231E12

Tangent value of a is: 1.0
```

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Java Math class: Methods

The **java.lang.Math** class contains various methods for performing basic numeric operations such as the logarithm, cube root, and trigonometric functions etc. The various java math methods are as follows:

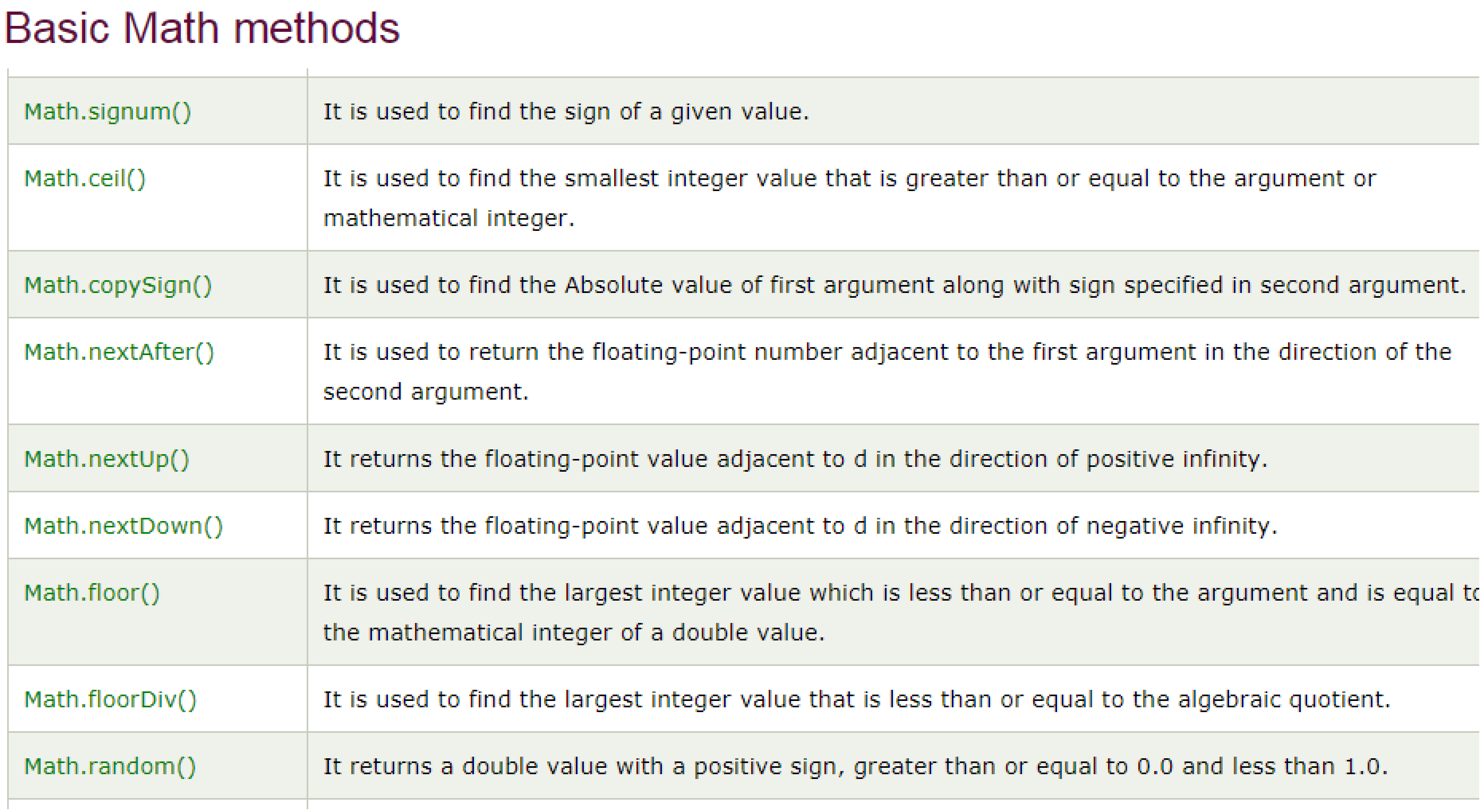
Basic Math methods



Method	Description
Math.abs()	It will return the Absolute value of the given value.
Math.max()	It returns the Largest of two values.
Math.min()	It is used to return the Smallest of two values.
Math.round()	It is used to round of the decimal numbers to the nearest value.
Math.sqrt()	It is used to return the square root of a number.
Math.cbrt()	It is used to return the cube root of a number.
Math.pow()	It returns the value of first argument raised to the power to second argument.

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Java Math class: Methods

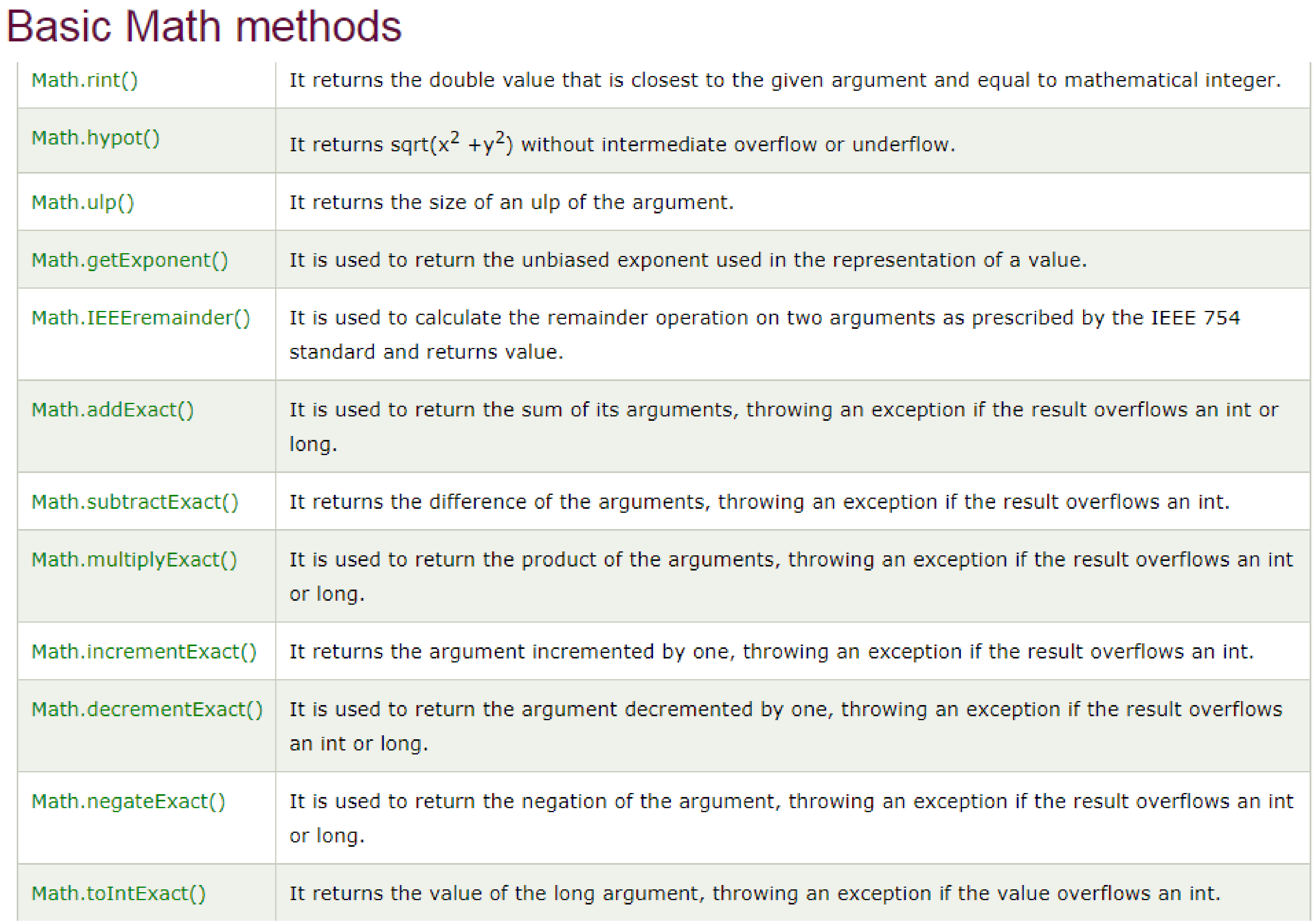


Basic Math methods

Math.signum()	It is used to find the sign of a given value.
Math.ceil()	It is used to find the smallest integer value that is greater than or equal to the argument or mathematical integer.
Math.copySign()	It is used to find the Absolute value of first argument along with sign specified in second argument.
Math.nextAfter()	It is used to return the floating-point number adjacent to the first argument in the direction of the second argument.
Math.nextUp()	It returns the floating-point value adjacent to d in the direction of positive infinity.
Math.nextDown()	It returns the floating-point value adjacent to d in the direction of negative infinity.
Math.floor()	It is used to find the largest integer value which is less than or equal to the argument and is equal t the mathematical integer of a double value.
Math.floorDiv()	It is used to find the largest integer value that is less than or equal to the algebraic quotient.
Math.random()	It returns a double value with a positive sign, greater than or equal to 0.0 and less than 1.0.

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

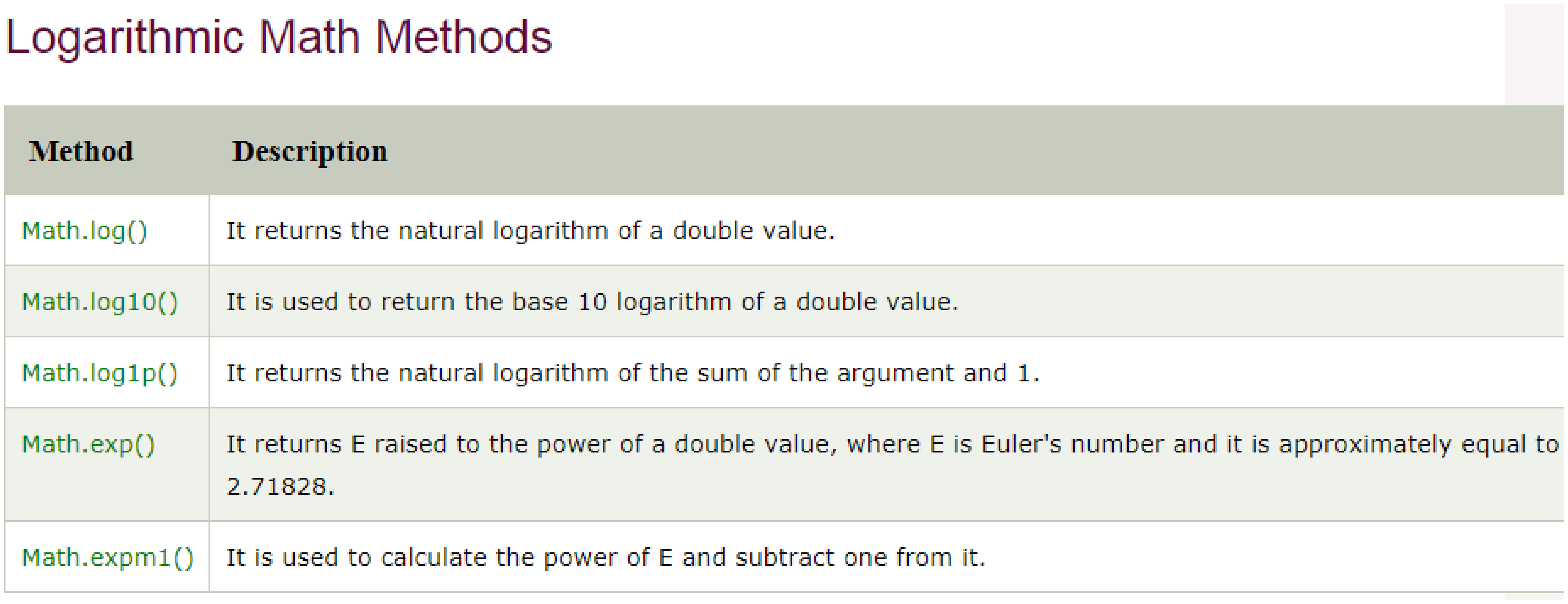
Java Math class: Methods



Basic Math methods

Math.rint()	It returns the double value that is closest to the given argument and equal to mathematical integer.
Math.hypot()	It returns sqrt(x^2 +y^2) without intermediate overflow or underflow.
Math.ulp()	It returns the size of an ulp of the argument.
Math.getExponent()	It is used to return the unbiased exponent used in the representation of a value.
Math.IEEEremainder()	It is used to calculate the remainder operation on two arguments as prescribed by the IEEE 754 standard and returns value.
Math.addExact()	It is used to return the sum of its arguments, throwing an exception if the result overflows an int or long.
Math.subtractExact()	It returns the difference of the arguments, throwing an exception if the result overflows an int.
Math.multiplyExact()	It is used to return the product of the arguments, throwing an exception if the result overflows an int or long.
Math.incrementExact()	It returns the argument incremented by one, throwing an exception if the result overflows an int.
Math.decrementExact()	It is used to return the argument decremented by one, throwing an exception if the result overflows an int or long.
Math.negateExact()	It is used to return the negation of the argument, throwing an exception if the result overflows an int or long.
Math.toIntExact()	It returns the value of the long argument, throwing an exception if the value overflows an int.

Java Math class: Methods

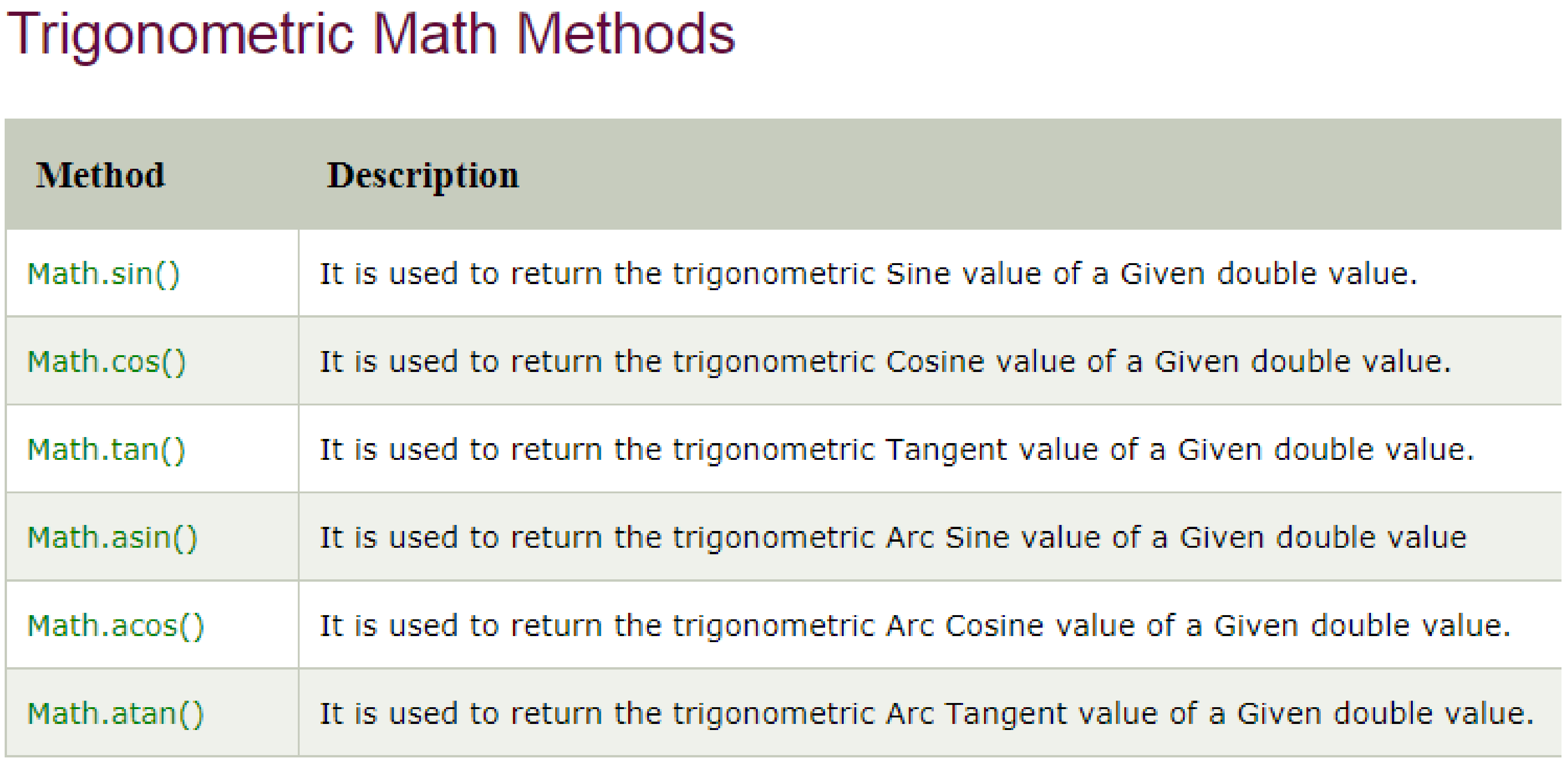


Logarithmic Math Methods

Method	Description
Math.log()	It returns the natural logarithm of a double value.
Math.log10()	It is used to return the base 10 logarithm of a double value.
Math.log1p()	It returns the natural logarithm of the sum of the argument and 1.
Math.exp()	It returns E raised to the power of a double value, where E is Euler's number and it is approximately equal to 2.71828.
Math.expm1()	It is used to calculate the power of E and subtract one from it.

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Java Math class: Methods

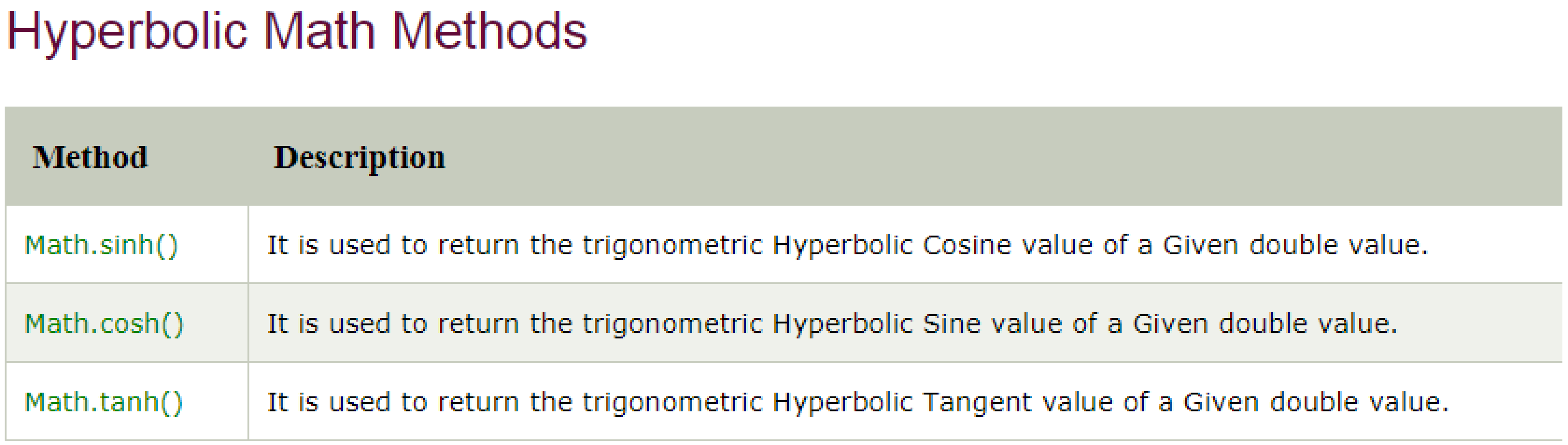


Trigonometric Math Methods

Method	Description
Math.sin()	It is used to return the trigonometric Sine value of a Given double value.
Math.cos()	It is used to return the trigonometric Cosine value of a Given double value.
Math.tan()	It is used to return the trigonometric Tangent value of a Given double value.
Math.asin()	It is used to return the trigonometric Arc Sine value of a Given double value
Math.acos()	It is used to return the trigonometric Arc Cosine value of a Given double value.
Math.atan()	It is used to return the trigonometric Arc Tangent value of a Given double value.

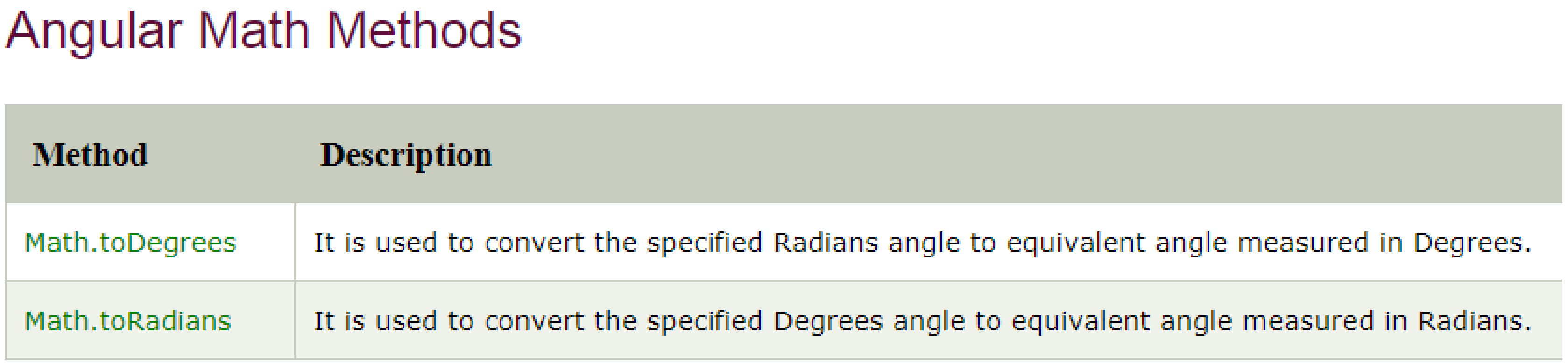
Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Java Math class: Methods



Hyperbolic Math Methods

Method	Description
Math.sinh()	It is used to return the trigonometric Hyperbolic Cosine value of a Given double value.
Math.cosh()	It is used to return the trigonometric Hyperbolic Sine value of a Given double value.
Math.tanh()	It is used to return the trigonometric Hyperbolic Tangent value of a Given double value.



Angular Math Methods

Method	Description
Math.toDegrees	It is used to convert the specified Radians angle to equivalent angle measured in Degrees.
Math.toRadians	It is used to convert the specified Degrees angle to equivalent angle measured in Radians.

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Java OOPs Miscellaneous

Assignment: Some of these concepts keeps as HW for the students

- Inner class
- Dynamic Binding
- Java Recursion
- Java Enum
- **Java Date and Time**
- Java Math class
- Wrapper classes – Auto boxing and Unboxing
- **Casting–Up-casting and Down-Casting**
- The Object Class in Java
- Object Cloning
- javadoc tool - Creating API Document

Java Date and Time

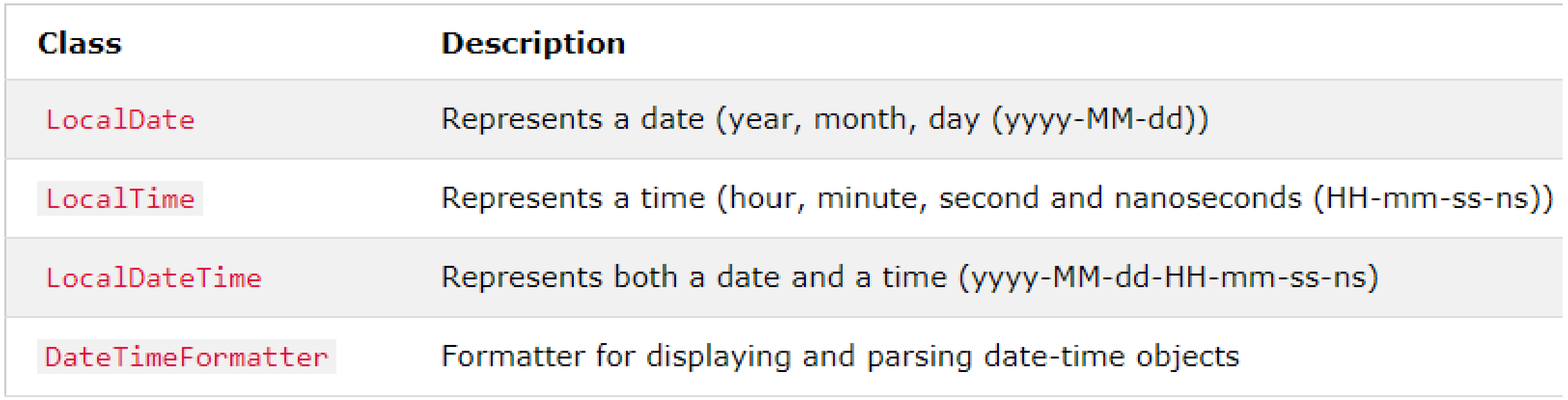
import the ***java.time*** package
for date and time API

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Java Date and Time

- **Java Date**

Java does not have a built-in Date class, but we can import the java.time package to work with the date and time API. The package includes many date and time classes. For example:



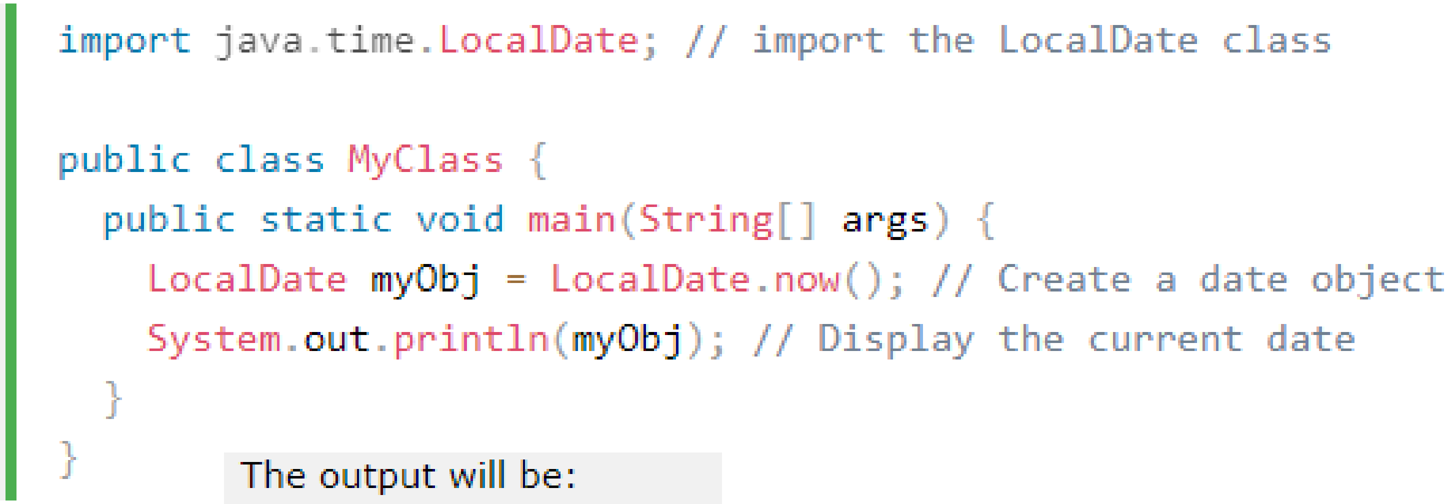
Class	Description
`LocalDate`	Represents a date (year, month, day (yyyy-MM-dd))
`LocalTime`	Represents a time (hour, minute, second and nanoseconds (HH-mm-ss-ns))
`LocalDateTime`	Represents both a date and a time (yyyy-MM-dd-HH-mm-ss-ns)
`DateTimeFormatter`	Formatter for displaying and parsing date-time objects

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

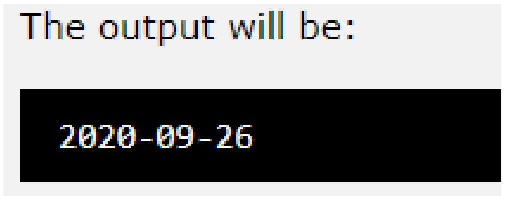
Java Date and Time

Display Current Date :

To display the current date, import the java.time.LocalDate class, and use its now() method:



```java
import java.time.LocalDate; // import the LocalDate class

public class MyClass {
  public static void main(String[] args) {
    LocalDate myObj = LocalDate.now(); // Create a date object
    System.out.println(myObj); // Display the current date
  }
}
```



The output will be:

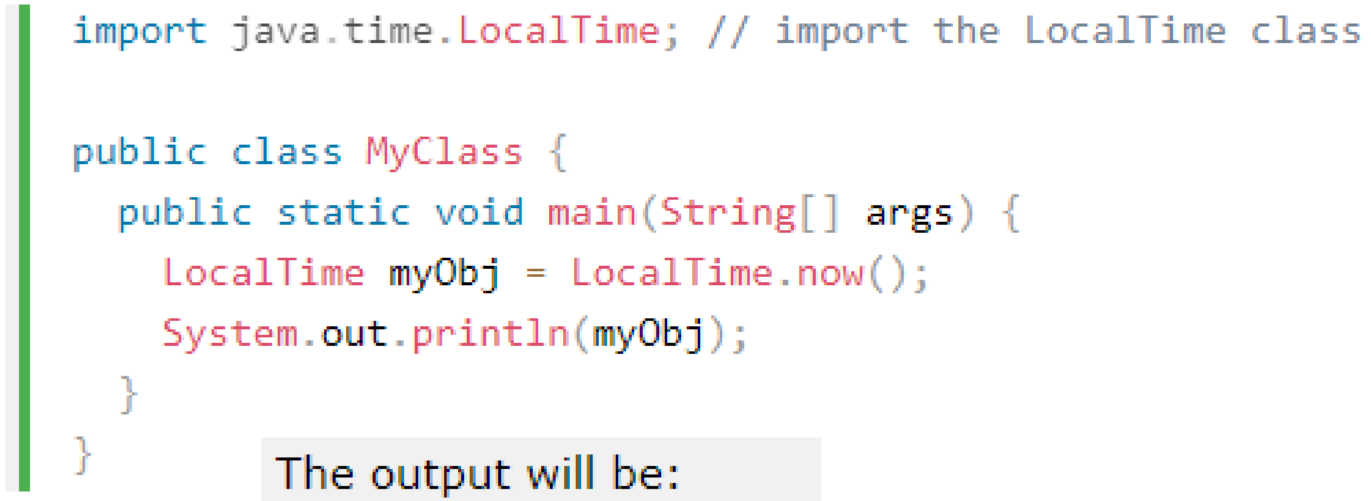
```
2020-09-26
```

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

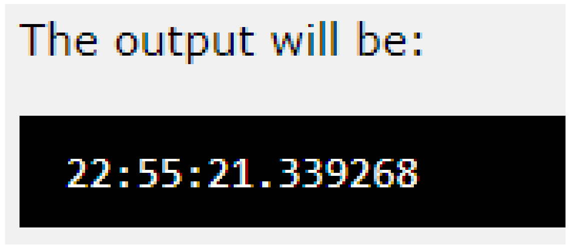
Java Date and Time

Display Current Time:

To display the current time (hour, minute, second, and nanoseconds), import the java.time.LocalTime class, and use its now() method:



```java
import java.time.LocalTime; // import the LocalTime class

public class MyClass {
  public static void main(String[] args) {
    LocalTime myObj = LocalTime.now();
    System.out.println(myObj);
  }
}
```



The output will be:

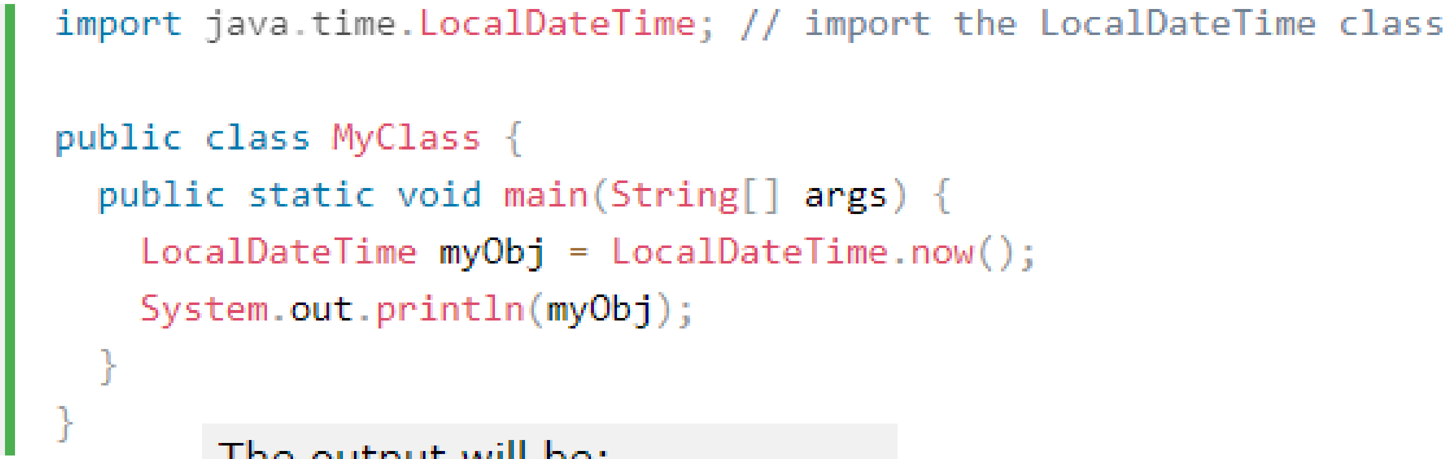
```
22:55:21.339268
```

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

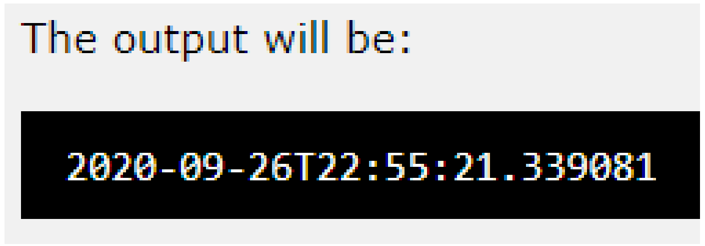
Java Date and Time

Display Current Date and Time

To display the current date and time, import the java.time.LocalDateTime class, and use its now() method:



```
import java.time.LocalDateTime; // import the LocalDateTime class

public class MyClass {
  public static void main(String[] args) {
    LocalDateTime myObj = LocalDateTime.now();
    System.out.println(myObj);
  }
}
```



The output will be:

```
2020-09-26T22:55:21.339081
```

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Java Date and Time

Formatting Date and Time

The "T" in the example above is used to separate the date from the time. You can use the DateTimeFormatter class with the ofPattern() method in the same package to format or parse date-time objects. The following example will remove both the "T" and nanoseconds from the date-time:

```java
import java.time.LocalDateTime; // Import the LocalDateTime class
import java.time.format.DateTimeFormatter; // Import the DateTimeFormatter class

public class MyClass {
  public static void main(String[] args) {
    LocalDateTime myDateObj = LocalDateTime.now();
    System.out.println("Before formatting: " + myDateObj);
    DateTimeFormatter myFormatObj = DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm:ss");

    String formattedDate = myDateObj.format(myFormatObj);
    System.out.println("After formatting: " + formattedDate);
  }
}
```

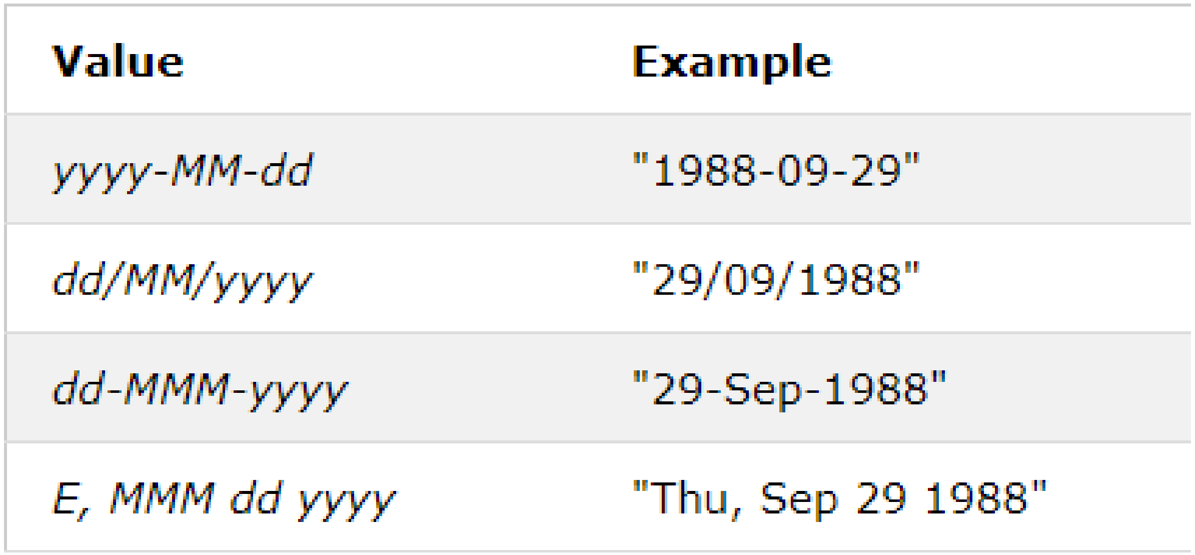
The output will be:

```
Before Formatting: 2020-09-26T22:55:21.340094
After Formatting: 26-09-2020 22:55:21
```

Java Date and Time

Formatting Date and Time

The ofPattern() method accepts all sorts of values, if you want to display the date and time in a different format. For example:



Value	Example
yyyy-MM-dd	"1988-09-29"
dd/MM/yyyy	"29/09/1988"
dd-MMM-yyyy	"29-Sep-1988"
E, MMM dd yyyy	"Thu, Sep 29 1988"

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Java OOPs Miscellaneous

Assignment: Some of these concepts keeps as HW for the students

- Inner class
- Dynamic Binding
- Java Recursion
- Java Enum
- Java Dates
- Java Math class
- **Wrapper classes – Auto boxing and Unboxing**
- Casting–Up-casting and Down-Casting
- The Object Class in Java
- Object Cloning
- javadoc tool - Creating API Document

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

OOP : Miscellaneous

Wrapper classes
Auto boxing and Unboxing

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Wrapper Class:
Auto boxing and Unboxing

Assignment: Hints

- *Wrapper classes – Auto boxing and Unboxing*

- **wrapper class**

The **wrapper class in Java** provides the mechanism *to convert primitive into object and object into primitive*.

- **Auto-boxing**

Auto-boxing is the automatic conversion that the Java compiler makes between the primitive types and their corresponding object wrapper classes. For example, converting an int to an Integer, a double to a Double, and so on.

- **Unboxing**

If the conversion goes the other way, this is called unboxing.

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Wrapper Class: Auto boxing and Unboxing

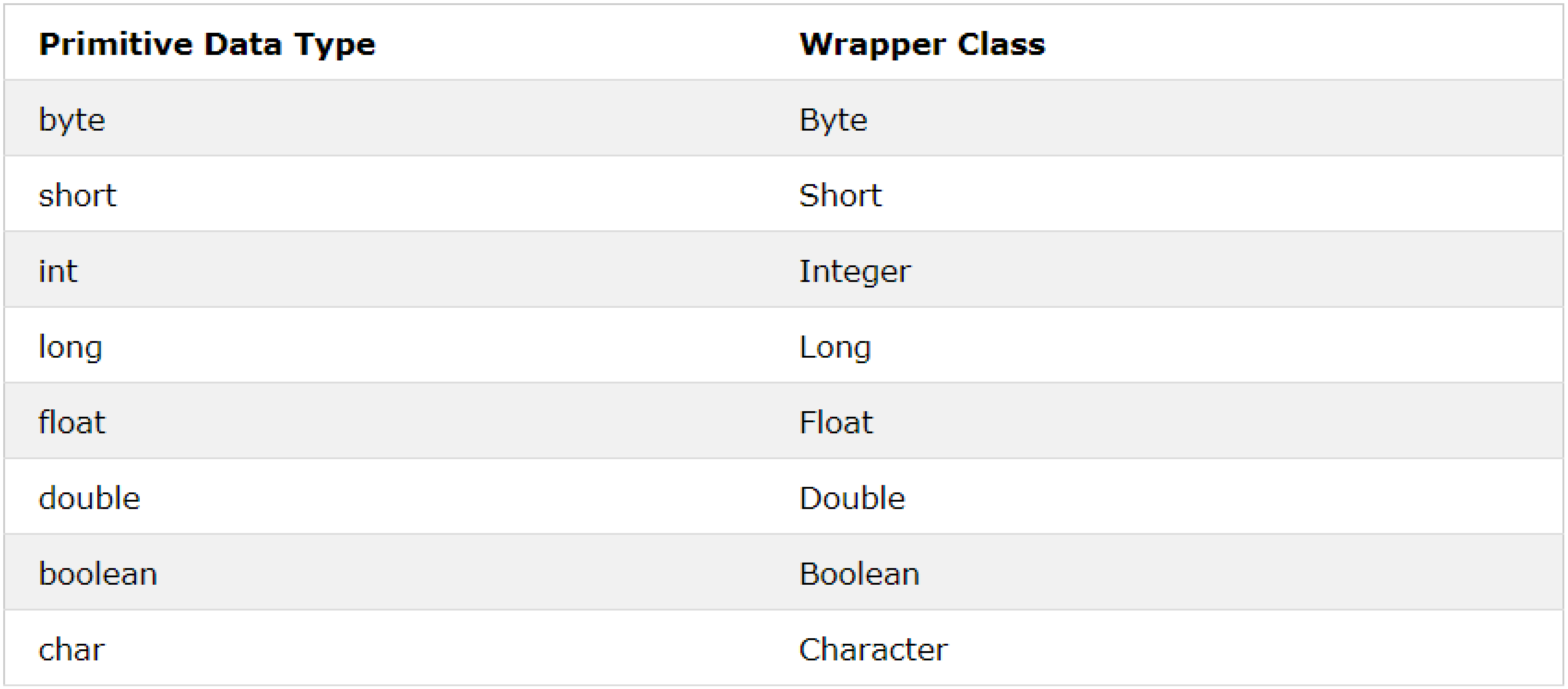
Use of Wrapper classes in Java

Java is an object-oriented programming language, so we need to deal with objects many times like in Collections, Serialization, Synchronization, etc.
Let us see the different scenarios, where we need to use the wrapper classes.

- **Change the value in Method:** Java supports only call by value. So, if we pass a primitive value, it will not change the original value. But, if we convert the primitive value in an object, it will change the original value.
- **Serialization:** We need to convert the objects into streams to perform the serialization. If we have a primitive value, we can convert it in objects through the wrapper classes.
- **Synchronization:** Java synchronization works with objects in Multithreading.
- **java.util package:** The java.util package provides the utility classes to deal with objects.
- **Collection Framework:** Java collection framework works with objects only. All classes of the collection framework (ArrayList, LinkedList, Vector, HashSet, LinkedHashSet, TreeSet, PriorityQueue, ArrayDeque, etc.) deal with objects only.

Wrapper Class: Auto boxing and Unboxing

The table below shows the primitive type and the equivalent wrapper class:



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

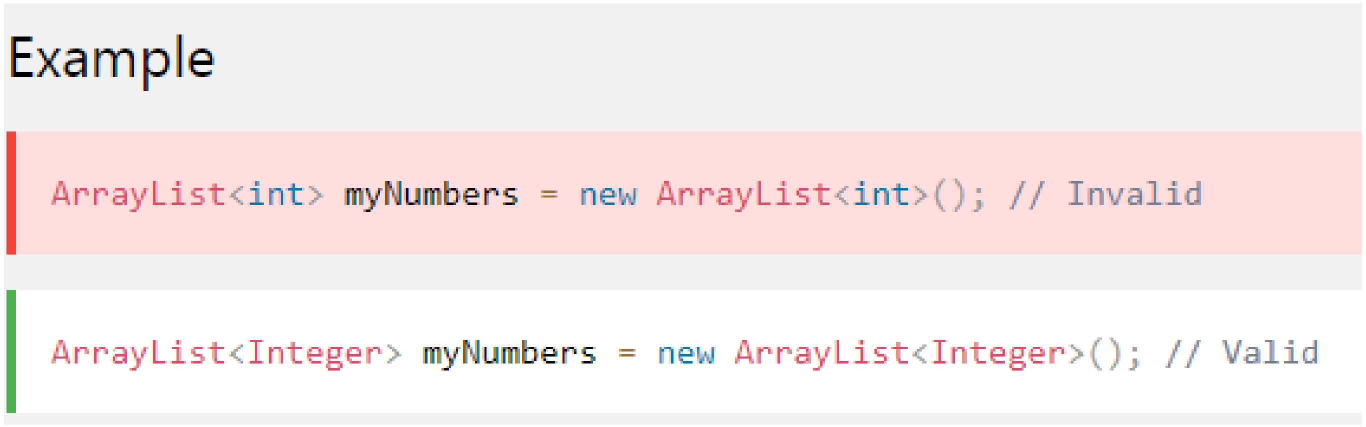
Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Wrapper Class: Auto boxing and Unboxing

Use of Wrapper classes in Java with Example

- **Collection Framework:**

Sometimes you must use wrapper classes, for example when working with Collection objects, such as ArrayList, where primitive types cannot be used (the list can only store objects):



Example

```
ArrayList<int> myNumbers = new ArrayList<int>(); // Invalid
```

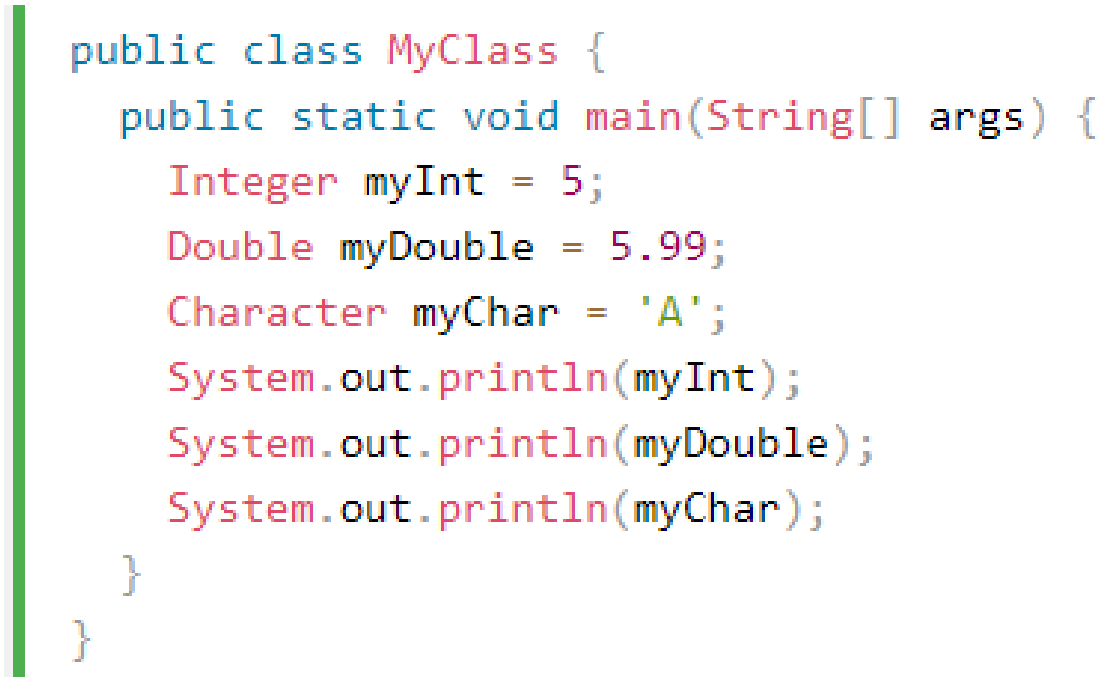
```
ArrayList<Integer> myNumbers = new ArrayList<Integer>(); // Valid
```

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Wrapper Class: Auto boxing and Unboxing

- **Creating Wrapper Objects**

To create a wrapper object, use the wrapper class instead of the primitive type. To get the value, you can just print the object:

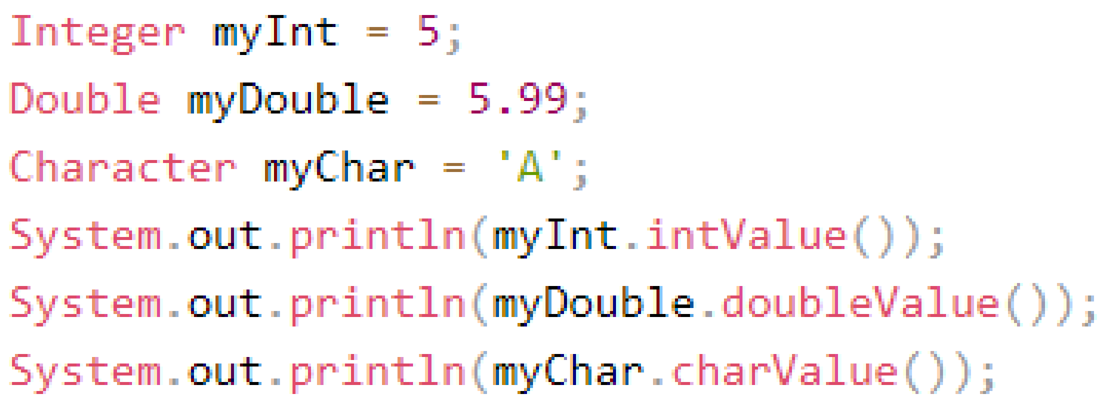


```
public class MyClass {
  public static void main(String[] args) {
    Integer myInt = 5;
    Double myDouble = 5.99;
    Character myChar = 'A';
    System.out.println(myInt);
    System.out.println(myDouble);
    System.out.println(myChar);
  }
}
```

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Wrapper Class:
Auto boxing and Unboxing

- Since you're now working with objects, you can use certain methods to get information about the specific object.
- For example, the following methods are used to get the value associated with the corresponding wrapper object: intValue(), byteValue(), shortValue(), longValue(), floatValue(), doubleValue(), charValue(), booleanValue().
- This example will output the same result as the example above:

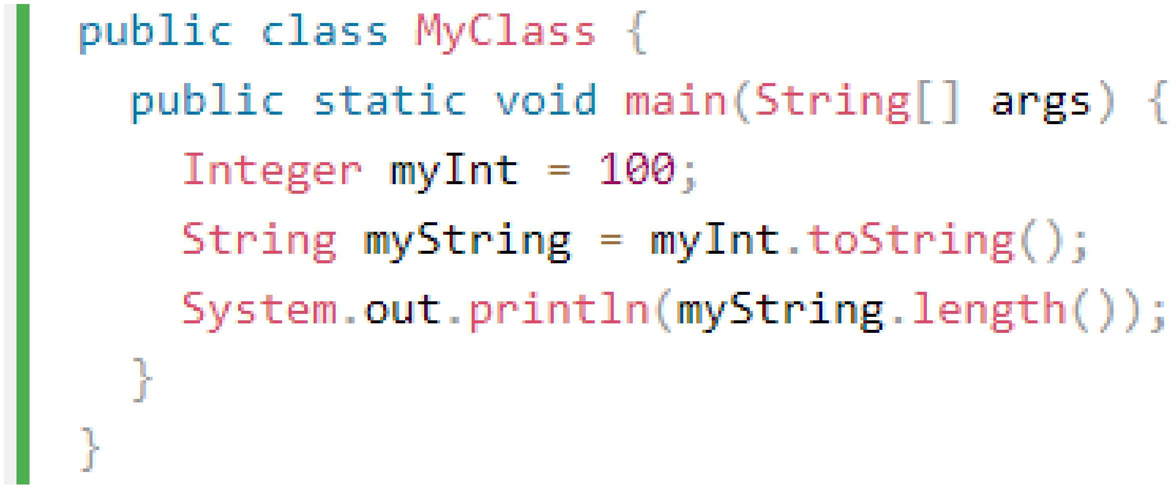


```
Integer myInt = 5;
Double myDouble = 5.99;
Character myChar = 'A';
System.out.println(myInt.intValue());
System.out.println(myDouble.doubleValue());
System.out.println(myChar.charValue());
```

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Wrapper Class: Auto boxing and Unboxing

- Another useful method is the toString() method, which is used to convert wrapper objects to strings.
- In the following example, we convert an Integer to a String, and use the length() method of the String class to output the length of the "string":



```
public class MyClass {
  public static void main(String[] args) {
    Integer myInt = 100;
    String myString = myInt.toString();
    System.out.println(myString.length());
  }
}
```

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Wrapper Class: Auto boxing

Auto-Boxing:

Primitive to Wrapper

The automatic conversion of primitive data type into its corresponding wrapper class is known as autoboxing

For example, byte to Byte, char to Character, int to Integer, long to Long, float to Float, boolean to Boolean, double to Double, and short to Short.

```
//Java program to convert primitive into objects
Autoboxing : int to Integer  (Primitive to Wrapper)
public class WrapperExample1{
public static void main(String args[]){
    //Converting int into Integer
     int a=20;
     //converting int into Integer explicitly
     Integer i=Integer.valueOf(a);
     //now compiler will  write Integer.valueOf(a)  internally
      Integer j=a;//autoboxing,
   System.out.println(a+" "+i+" "+j);
  }
}
```

Output: 20 20 20

Since Java 5, we do not need to use the valueOf() method of wrapper classes to convert the primitive into objects.

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Wrapper Class: Unboxing

Unboxing:
Wrapper to Primitive
The automatic conversion of wrapper type into its corresponding primitive type is known as unboxing.

It is the reverse process of autoboxing.

```
//Java program to convert object into primitives
//Unboxing : Integer to int  (Wrapper to Primitive)
public class WrapperExample2{
public static void main(String args[]){
        //Converting Integer to int
         Integer a=new Integer(3);
        //converting Integer to int explicitly
           int i=a.intValue();
         // now compiler will write a.intValue() internally
         int j=a;//unboxing,

        System.out.println(a+" "+i+" "+j);
}}
```

Output: 3 3   3

Since Java 5, we do not need to use the intValue() method of wrapper classes to convert the wrapper type into primitives.

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Wrapper Class: Auto boxing and Unboxing

Summary:

autoboxing and unboxing feature convert primitives into objects and objects into primitives automatically.

The automatic conversion of primitive into an object is known as autoboxing and vice-versa unboxing.

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Java OOPs Miscellaneous

Assignment: Some of these concepts keeps as HW for the students

- Inner class
- Dynamic Binding
- Java Recursion
- Java Enum
- Java Dates
- Java Math class
- Wrapper classes – Auto boxing and Unboxing
- **Casting–Up-casting and Down-Casting**
- The Object Class in Java
- Object Cloning
- javadoc tool - Creating API Document

OOP : Miscellaneous

Casting

Up-casting and Down-Casting

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Typecasting

Typecasting : Object Type

Typecasting is one of the most important concepts which basically deals with the conversion of one data type to another datatype implicitly or explicitly. In this article, the concept of the typecasting for objects is discussed.

Just like the datatypes, the objects can also be typecasted. However, in objects, there are only two types of objects (i.e.) parent object and child object. Therefore, typecasting of objects basically mean that one type of object (i.e.) child or parent to another. There are two types of typecasting.

They are: **Upcasting** and **Downcasting**

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Up-casting and Down-Casting

Object Typecasting : Category

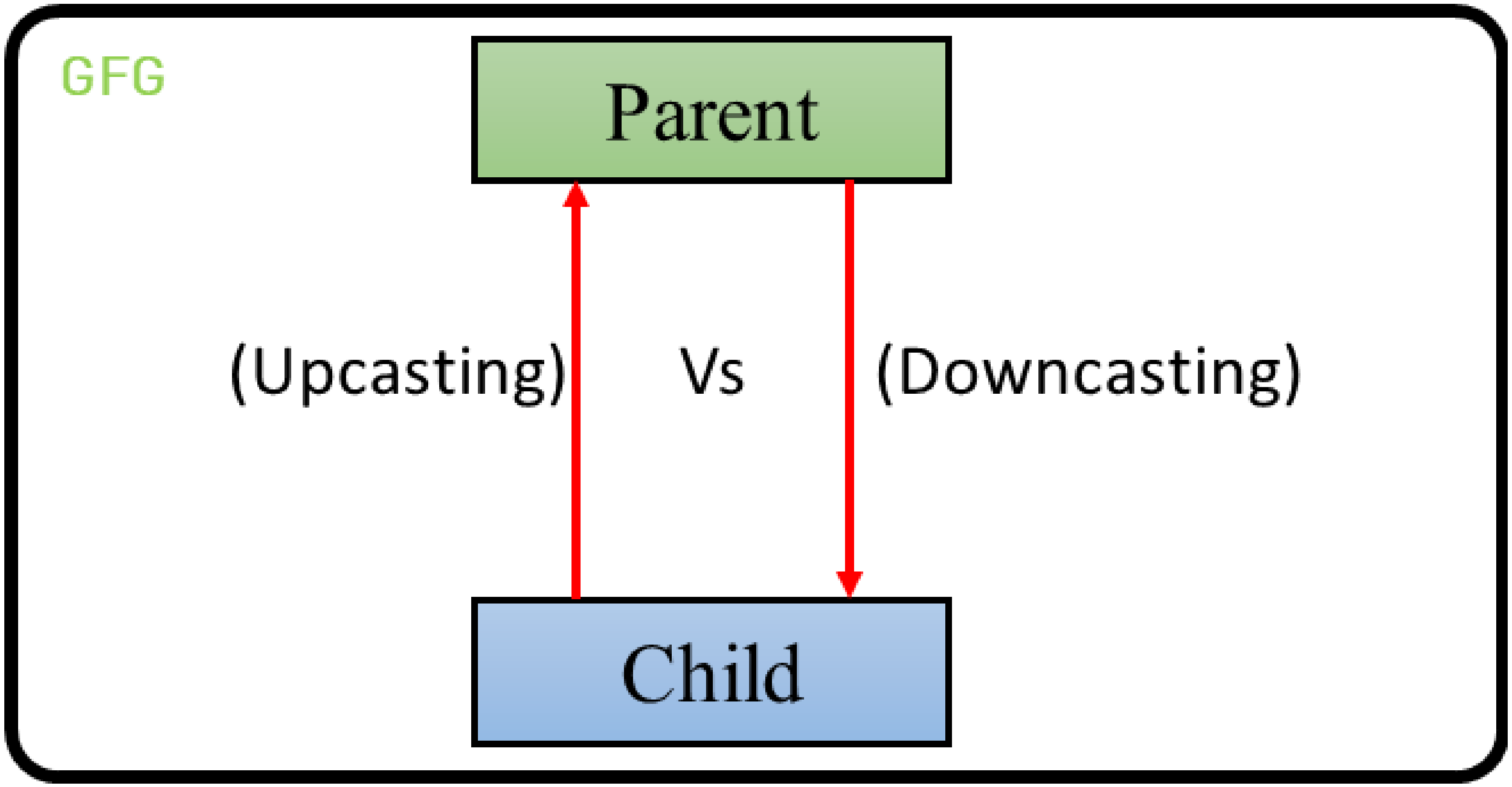
- **Upcasting**: Upcasting is the typecasting of a child object to a parent object. *Upcasting can be done implicitly*.

Upcasting gives us the flexibility to access the parent class members but it is not possible to access all the child class members using this feature. Instead of all the members, we can access some specified members of the child class. For instance, we can access the overridden methods.

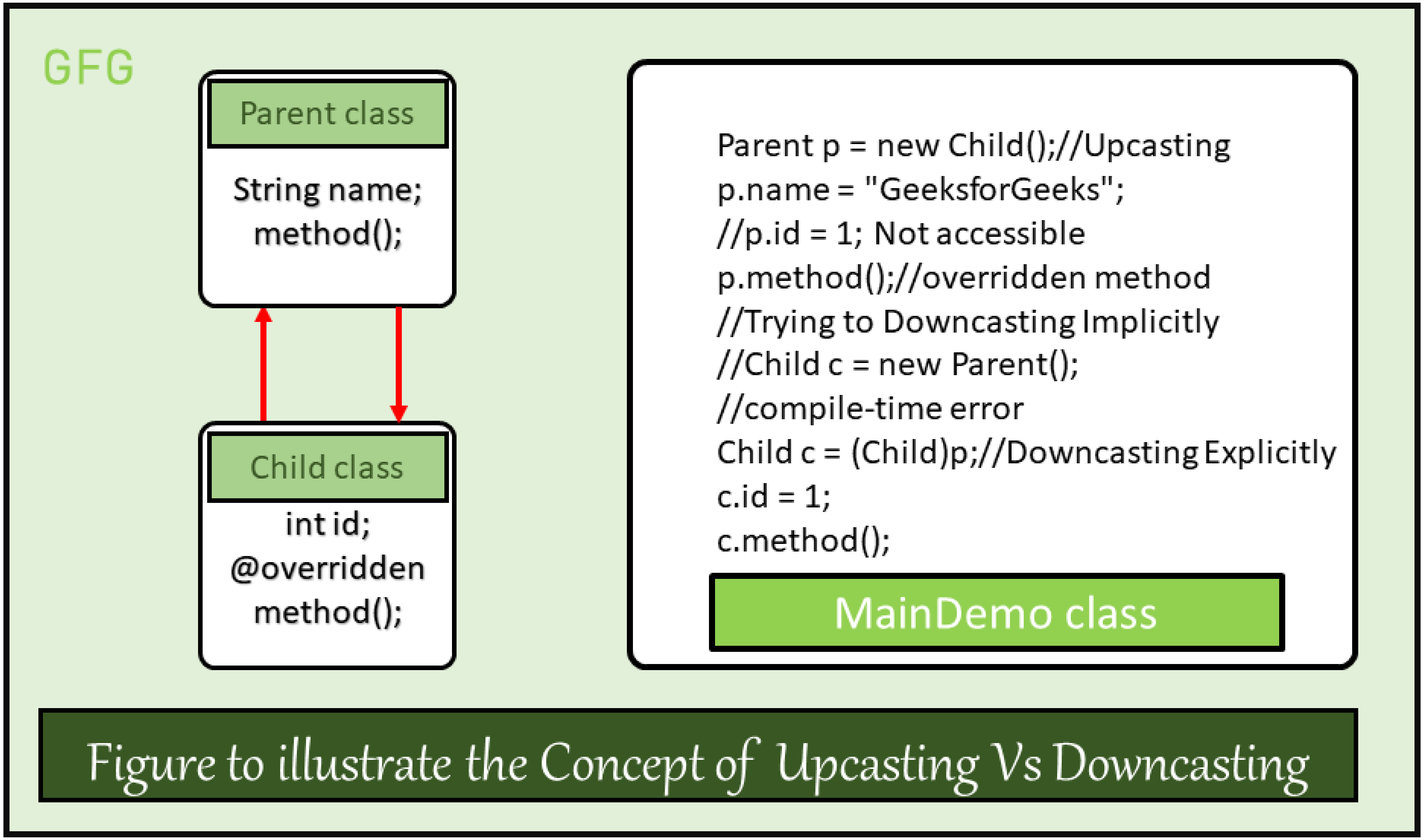
- **Downcasting**: Similarly, downcasting means the typecasting of a parent object to a child object. *Downcasting cannot be implicitly*.

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Upcasting and Downcasting



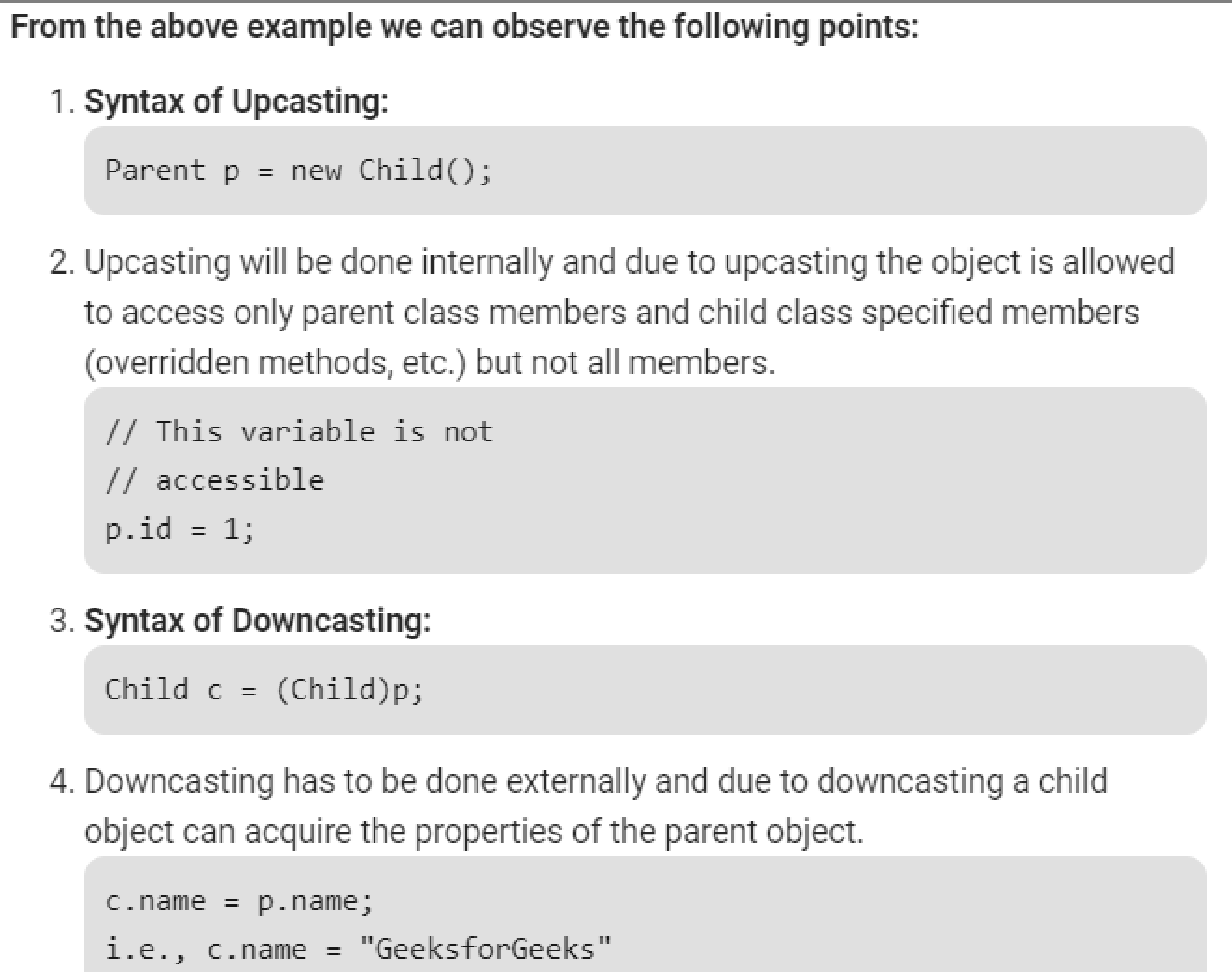
Upcasting and Downcasting



```
Parent p = new Child();//Upcasting
p.name = "GeeksforGeeks";
//p.id = 1; Not accessible
p.method();//overridden method
//Trying to Downcasting Implicitly
//Child c = new Parent();
//compile-time error
Child c = (Child)p;//Downcasting Explicitly
c.id = 1;
c.method();
```

Figure to illustrate the Concept of Upcasting Vs Downcasting

Upcasting and Downcasting



From the above example we can observe the following points:

1. **Syntax of Upcasting:**

```
Parent p = new Child();
```

2. Upcasting will be done internally and due to upcasting the object is allowed to access only parent class members and child class specified members (overridden methods, etc.) but not all members.

```
// This variable is not
// accessible
p.id = 1;
```

3. **Syntax of Downcasting:**

```
Child c = (Child)p;
```

4. Downcasting has to be done externally and due to downcasting a child object can acquire the properties of the parent object.

```
c.name = p.name;
i.e., c.name = "GeeksforGeeks"
```

Java OOPs Miscellaneous

Assignment: Some of these concepts keeps as HW for the students

- Inner class
- Dynamic Binding
- Java Recursion
- Java Enum
- Java Dates
- Java Math class
- Wrapper classes – Auto boxing and Unboxing
- Casting–Up-casting and Down-Casting
- **The Object Class in Java**
- Object Cloning
- javadoc tool - Creating API Document

OOP : Miscellaneous

The Object Class

Parent class of all classes by default

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

The Object Class

- The Object class is the parent class of all the classes in java by default. In other words, it is the topmost class of java.

- The Object class is beneficial if you want to refer any object whose type you don't know.
  - *Notice that parent class reference variable can refer the child class object, know as upcasting.*

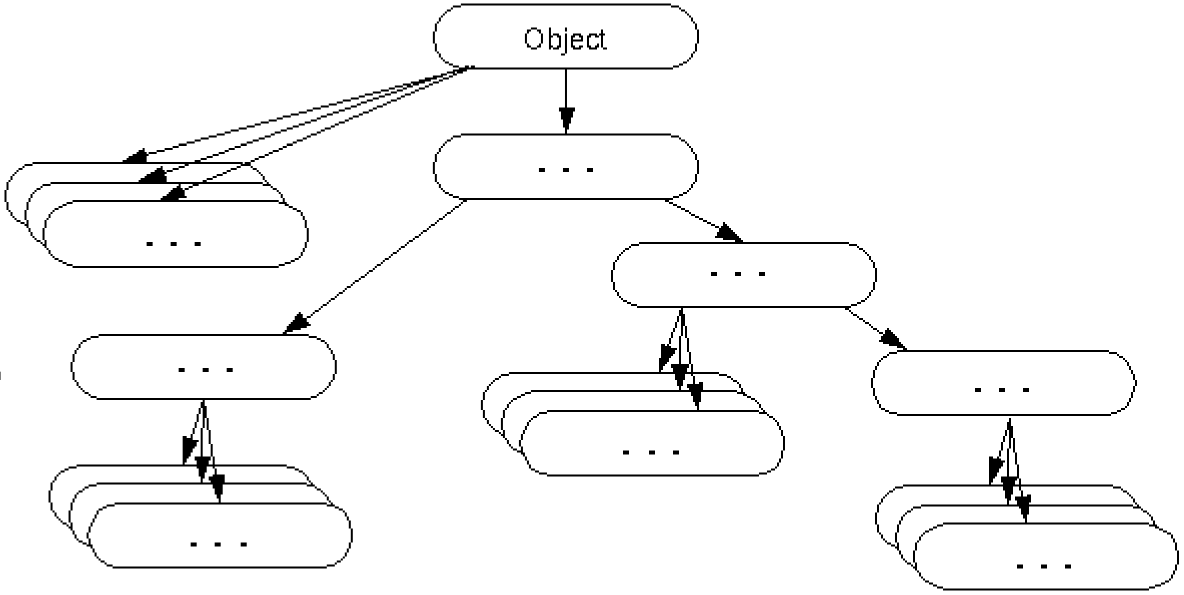
    - *As opposed to downcasting or type refinement is the act of casting a reference of a base class to one of its derived classes*

The Object Class: Example

- There is getObject() method that returns an object but it can be of any type like Employee, Student etc., we can use Object class reference to refer that object. For example:

Object obj=getObject();//we don't know what object will be returned from this method

The Object class provides some common behaviors to all the objects *such as object can be* ***compared****, objec can be* ***cloned****, object can be* **notified** *etc.*



The Object Class: Example

- **Object class Summary**

  - Object class is present in java.lang package.
  - Every class in Java is directly or indirectly derived from the Object class.
  - If a Class does not extend any other class then it is direct child class of Object and if extends other class then it is an indirectly derived.

Therefore the Object class methods are available to all Java classes. Hence Object class acts as a root of inheritance hierarchy in any Java Program.

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Java OOPs Miscellaneous

Assignment: Some of these concepts keeps as HW for the students

- Inner class
- Dynamic Binding
- Java Recursion
- Java Enum
- Java Dates
- Java Math class
- Wrapper classes – Auto boxing and Unboxing
- The Object Class in Java
- **Object Cloning**
- javadoc tool - Creating API Document

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

OOP : Miscellaneous

Object Cloning

create exact copy of an object

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Object Cloning

- The **object cloning** is a way to create exact copy of an object. The clone() method of Object class is used to clone an object.

- The **java.lang.Cloneable interface** must be implemented by the class whose object clone we want to create.

- If we don't implement Cloneable interface, clone() method generates **CloneNotSupportedException**.

The **clone() method** is defined in the Object class. Syntax of the clone() method is as follows:

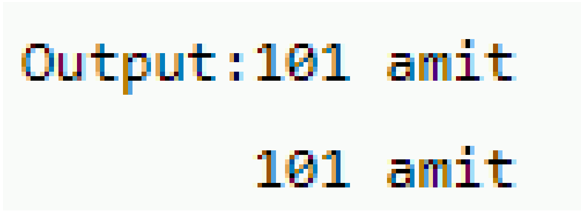
protected *Object clone()* ***throws*** *CloneNotSupportedException*

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Object Cloning : Example

```
class Student implements Cloneable{
        int rollno;
        String name;

        Student(int rollno,String name){
                this.rollno=rollno;
                this.name=name;
        }

        public Object clone()
        throws CloneNotSupportedException{
        return super.clone();
        }
```

```
public static void main(String args[]){
  try{
        Student s1=new Student(101,"amit");

        Student s2=(Student)s1.clone();

        System.out.println(s1.rollno+" "+s1.name);
        System.out.println(s2.rollno+" "+s2.name);
        }
   catch(CloneNotSupportedException c){}
}

}
```



```
Output:101 amit

       101 amit
```

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Object Cloning

- **Why use clone() method ?**

The clone() method saves the extra processing task for creating the exact copy of an object. If we perform it by using the new keyword, it will take a lot of processing time to be performed that is why we use object cloning.

Object Cloning

Advantage of Object cloning

Although Object.clone() has some design issues but it is still a popular and easy way of copying objects.

Following is a list of advantages of using clone() method:

- You don't need to write lengthy and repetitive codes. Just use an abstract class with a 4- or 5-line long clone() method.
- It is the easiest and most efficient way for copying objects, especially if we are applying it to an already developed or an old project. Just define a parent class, implement Cloneable in it, provide the definition of the clone() method and the task will be done.
- Clone() is the fastest way to copy array.

Object Cloning

Advantage of Object cloning

Although Object.clone() has some design issues but it is still a popular and easy way of copying objects.

Following is a list of advantages of using clone() method:

- You don't need to write lengthy and repetitive codes. Just use an abstract class with a 4- or 5-line long clone() method.
- It is the easiest and most efficient way for copying objects, especially if we are applying it to an already developed or an old project. Just define a parent class, implement Cloneable in it, provide the definition of the clone() method and the task will be done.
- Clone() is the fastest way to copy array.

Object Cloning

Disadvantage of Object cloning

Following is a list of some disadvantages of clone() method:

- To use the Object.clone() method, we have to change a lot of syntaxes to our code, like implementing a Cloneable interface, defining the clone() method and handling CloneNotSupportedException, and finally, calling Object.clone() etc.
- We have to implement cloneable interface while it doesn't have any methods in it. We just have to use it to tell the JVM that we can perform clone() on our object.
- Object.clone() is protected, so we have to provide our own clone() and indirectly call Object.clone() from it.
- Object.clone() doesn't invoke any constructor so we don't have any control over object construction.
- If you want to write a clone method in a child class then all of its superclasses should define the clone() method in them or inherit it from another parent class. Otherwise, the super.clone() chain will fail.
- Object.clone() supports only shallow copying but we will need to override it if we need deep cloning.

Java OOPs Miscellaneous

Assignment: Some of these concepts keeps as HW for the students

- Inner class
- Dynamic Binding
- Java Recursion
- Java Enum
- Java Dates
- Java Math class
- Wrapper classes – Auto boxing and Unboxing
- The Object Class in Java
- Object Cloning
- **javadoc tool - Creating API Document**

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

OOP : Miscellaneous

javadoc tool

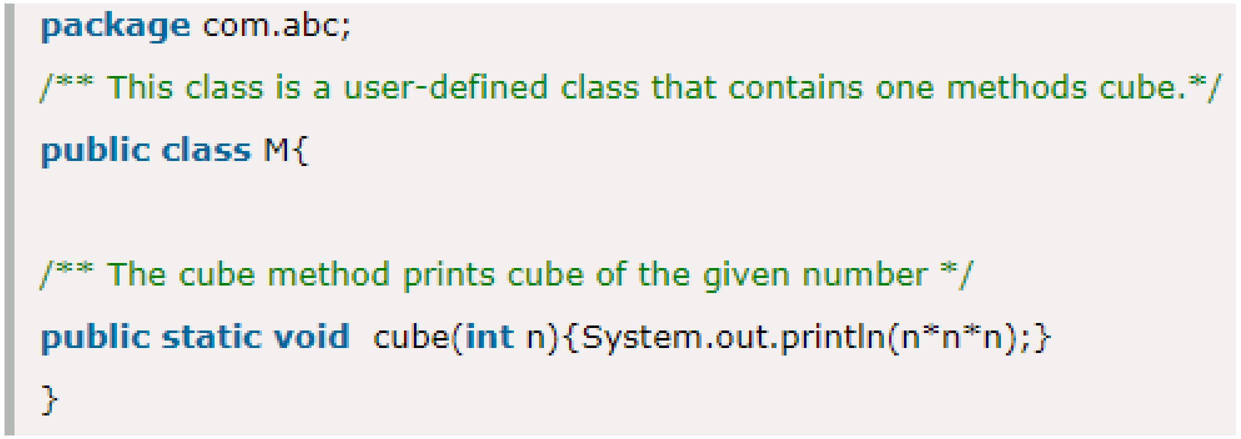
Creating API Document

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

javadoc tool

- We can create document api in java by the help of **javadoc** tool.
- In the java file, we must use the documentation comment /**... */ to post information for the class, method, constructor, fields etc.

Let's see the simple class that contains documentation comment.



```java
package com.abc;
/** This class is a user-defined class that contains one methods cube.*/
public class M{

/** The cube method prints cube of the given number */
public static void  cube(int n){System.out.println(n*n*n);}
}
```

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

javadoc tool

- To create the document API, you need to use the javadoc tool followed by java file name. There is no need to compile the javafile.
- On the command prompt, you need to write:

```
javadoc  M.java
```

to generate the document api.

Now, there will be created a lot of html files. Open the index.html file to get the information about the classes.