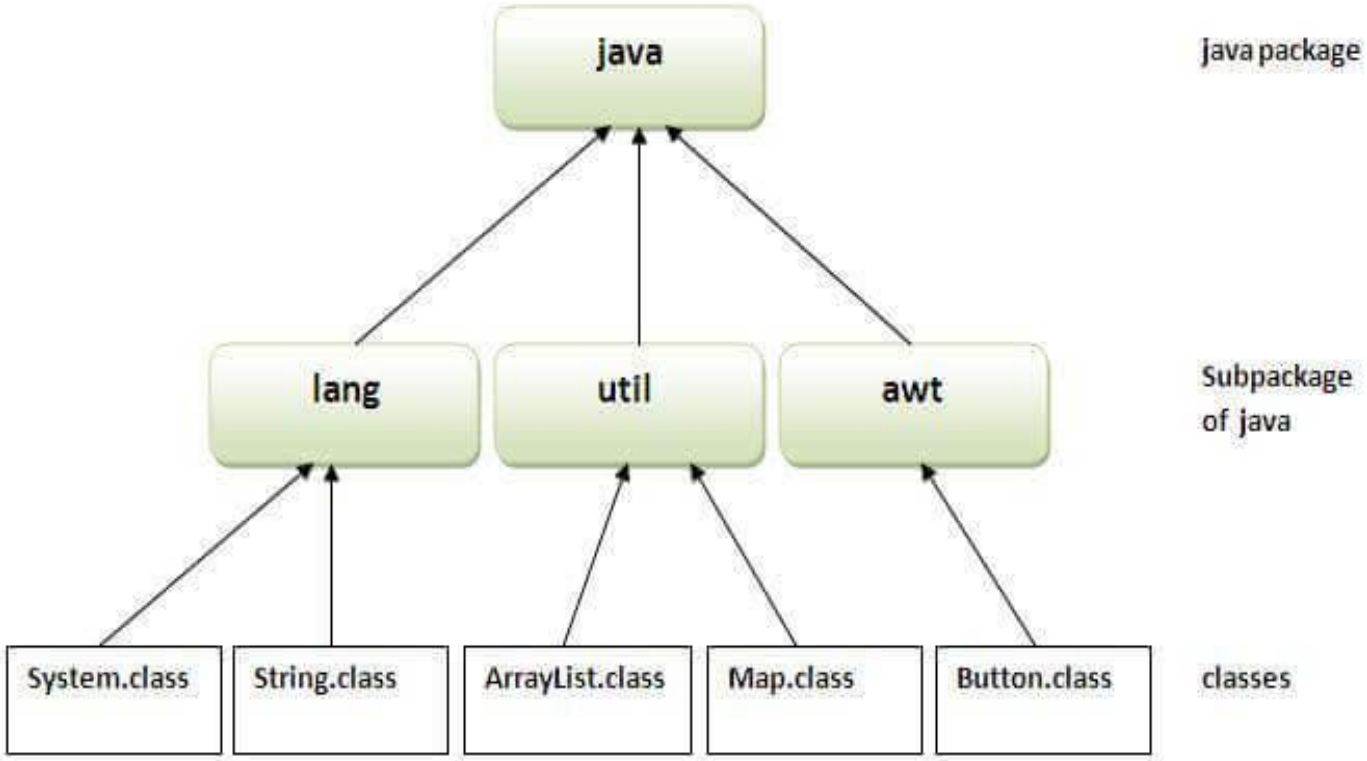
Unit 9
Understanding Core Java Packages

Package in Java

- A java package is a group of similar types of classes, interfaces and sub-packages.
- Package in java can be categorized in two form, built-in package and user-defined package.
- There are many built-in packages such as java, lang, awt, javax, swing, net, io, util, sql etc.

Package In Java



java.lang Package

- The package java.lang contains classes and interfaces that are essential to the Java language.
- These include:
  - Object, the ultimate superclass of all classes in Java
  - Thread, the class that controls each thread in a multithreaded program
  - Throwable, the superclass of all error and exception classes in Java
  - Integer, Float, Double etc., Classes that encapsulate the primitive data types in Java
  - Math, a class that provides standard mathematical methods
  - String, the class that is used to represent strings
- Because the classes in the java.lang package are so essential, the java.lang package is implicitly imported by every Java source file.

Java.lang.Math Class

- Math Class methods helps to perform the numeric operations like square, square root, cube, cube root, exponential and trigonometric operations
- Methods of Math class
  1. Basic Math Methods
  2. Logarithmic Math Methods
  3. Trigonometric Math Methods
  4. Hyperbolic Math Methods
  5. Angular Math Methods

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.
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 to the mathematical integer of a double value.

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.

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.

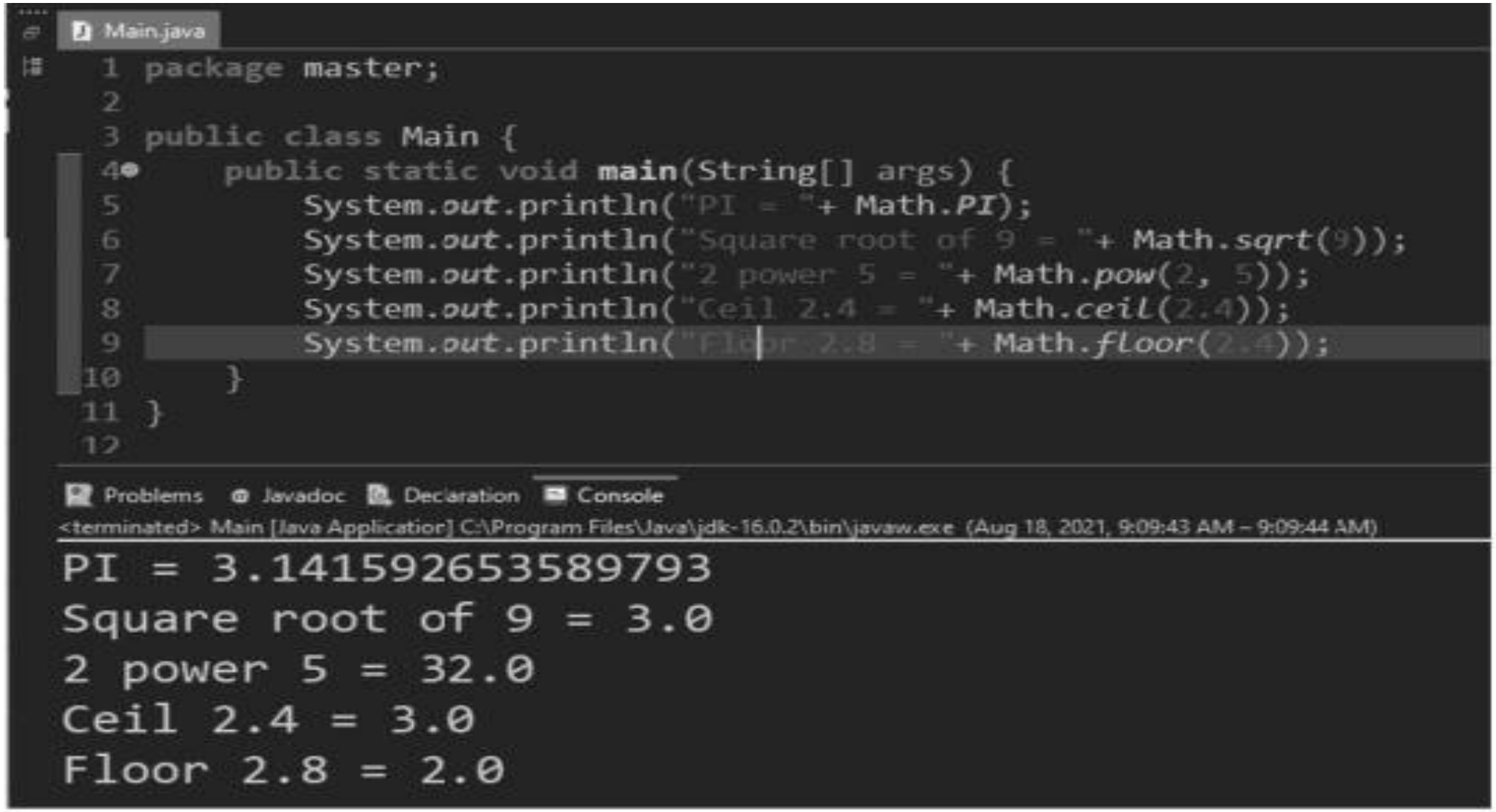
Hyperbolic Math Methods

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.

Java.lang.Math Example



Wrapper Class in Java

- A Wrapper class is a class whose object wraps or contains primitive data types.
- The wrapper class in Java provides the mechanism to convert primitive data types (int, boolean, etc) into object and object into primitive type
- **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 is unboxing.

Wrapper Class in Java

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

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

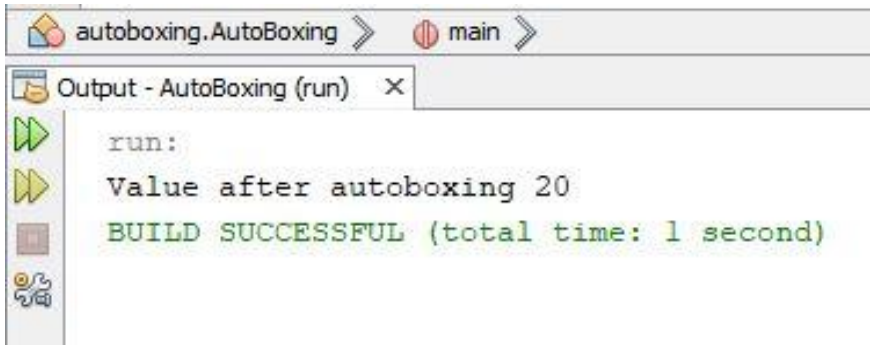
Autoboxing In Java

- 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.
- Since Java 5, we do not need to use the valueOf() method for autoboxing.

```
public class AutoBoxing {

    public static void main(String[] args) {
        int num = 20;
        Integer integerNum = Integer.valueOf(num);  //autoboxing
        System.out.println("Value after autoboxing "+integerNum);
    }

}
```



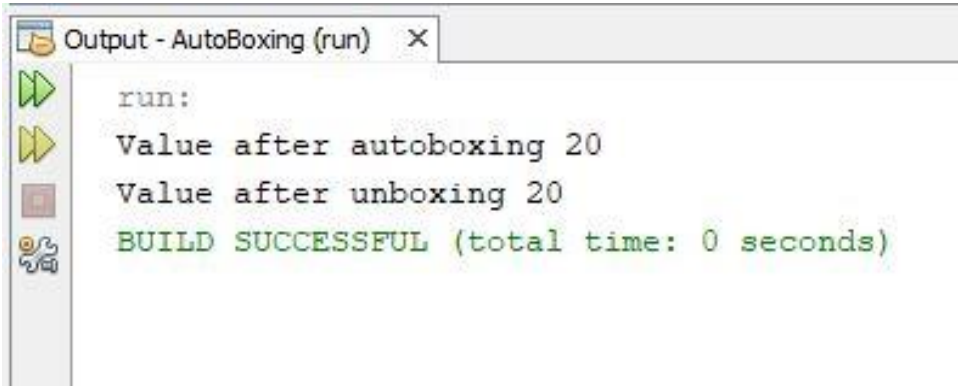
Unboxing in Java

- The automatic conversion of wrapper type into its corresponding primitive type is known as unboxing.
- It is the reverse process of autoboxing.

```
public class AutoBoxing {

    public static void main(String[] args) {
        int num = 20;
        Integer integerNum = Integer.valueOf(num);  //autoboxing
        System.out.println("Value after autoboxing "+integerNum);
        int intNum = integerNum; //unboxing
        System.out.println("Value after unboxing "+intNum);
    }

}
```



Advantages of Wrapper Classes

- Collection Framework: Java collection framework works with objects only
- 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.
- We can store the null value in wrapper objects
- Serialization: We need to convert the objects into streams to perform the serialization.
- Synchronization: Java synchronization works with objects in Multithreading.
- java.util package: The java.util package provides the utility classes to deal with objects.
- Note: Primitive types are more efficient than corresponding objects. Hence, when efficiency is the requirement, it is always recommended primitive types

java.util package

- It contains the collections framework, legacy collection classes, event model, date and time facilities, internationalization, and miscellaneous utility classes (a string tokenizer, a random-number generator, and a bit array)
- Following are some of the Important Classes in Java.util package:
  - ArrayList: Resizable-array implementation of the List interface.
  - Arrays: This class contains various methods for manipulating arrays (such as sorting and searching)
  - Date: The class Date represents a specific instant in time, with millisecond precision.
  - Dictionary: The Dictionary class is the abstract parent of any class, such as Hashtable, which maps keys to values.
  - Vector: The Vector class implements a growable array of objects.

java.util package

- HashMap: Hash table based implementation of the Map interface.
- LinkedList: Doubly-linked list implementation of the List and Deque interfaces.
- PriorityQueue: An unbounded priority queue based on a priority heap.
- Random: An instance of this class is used to generate a stream of pseudorandom numbers.
- Scanner: A simple text scanner which can parse primitive types and strings using regular expressions.
- Stack: The Stack class represents a last-in-first-out (LIFO) stack of objects.
- TreeSet: A NavigableSet implementation based on a TreeMap.
- EnumMap: A specialized Map implementation for use with enum type keys.
- EnumSet: A specialized Set implementation for use with enum types.

Enumerations

- In Java, Enumerations (enum in short) is a type that has a fixed set of constant values.
- We use enum keyword to declare enums

```
enum Size{
    SM,MD,LG,XL
}
```

- Above is an enum named size that contains fixed values SM, MD,LG and XL.
- The values inside the braces are called enum constants.
- The enum constants are usually represented in uppercase.

Enumerations-Example

```
enum Size{
    SM,MD,LG,XL
}

public class Main {
    public static void main(String[] args) {
        Size shirtSize = Size.SM;
        System.out.println("Shirt Size:"+shirtSize);
    }
}
```

```
Shirt Size:SM
```

Random Number Generation

- Java provides two main ways to generate random numbers using some built-in methods and classes as listed below:
  1. java.util.Random class
  2. Math.random method : Can Generate Random Numbers of double type.

  Note: There are also other methods to generate random number

1. java.util.Random Class

- For using this class to generate random numbers, we have to first create an instance of this class and then invoke methods such as nextInt(), nextDouble(), nextLong() etc using that instance.
- We can pass arguments to the methods for placing an upper bound on the range of the numbers to be generated. For example, nextInt(6) will generate numbers in the range 0 to 5 both inclusive

Random Number Generation

```java
import java.util.Random;

public class Main {
    public static void main(String[] args) {
        Random r = new Random();
        int n1 = r.nextInt(6);
        int n2 = r.nextInt(6);
        System.out.println("Random Integer: "+n1);
        System.out.println("Random Integer: "+n2);

        double d1 = r.nextDouble();
        double d2 = r.nextDouble();
        System.out.println("Random Double: "+d1);
        System.out.println("Random Double: "+d2);
    }
}
```

```
Random Integer: 0
Random Integer: 2
Random Double: 0.6806022470379256
Random Double: 0.6075298899263222
```

Random Number Generation

Math.random()

- This method returns a double value with a positive sign, greater than or equal to 0.0 and less than 1.0
- This method can only generate random numbers of type Doubles.
- We can use the following formula to generate a random number between a specified range.

Math.random() * (max - min + 1) + min

```
System.out.println("Random 1: "+Math.random());
System.out.println("Random 2: "+Math.random());
```

```
Random 1: 0.8453060056698111
Random 2: 0.5756310760667458
```