Learning Java - A Foundational Journey

Session: 15

Additional Features of Java

Objectives

- List some of the deprecated or removed features between Java 9 to 15
- Explain the new mathematical functions in Java
- Explain Java modules
- Describe hidden and sealed classes

Deprecated Features in Recent Versions of Java 1-2

Removal of constructors of classes corresponding to primitive types

- Classes such as Boolean, Byte, Short, Character, Integer, Long, Float, and Double are considered as 'box' classes, because they can use autoboxing

Removal of Nashorn Scripting engine and jjs tool

- Nashorn was the JavaScript scripting engine introduced in Java 8
- The main goal of Nashorn was to provide a lightweight and high-performance JavaScript runtime in Java with a native JVM

Internal garbage collector options

- Several garbage collector options have been removed (such as Concurrent Mark Sweep (CMS) Garbage Collector)

Deprecated Features in Recent Versions of Java 2-2

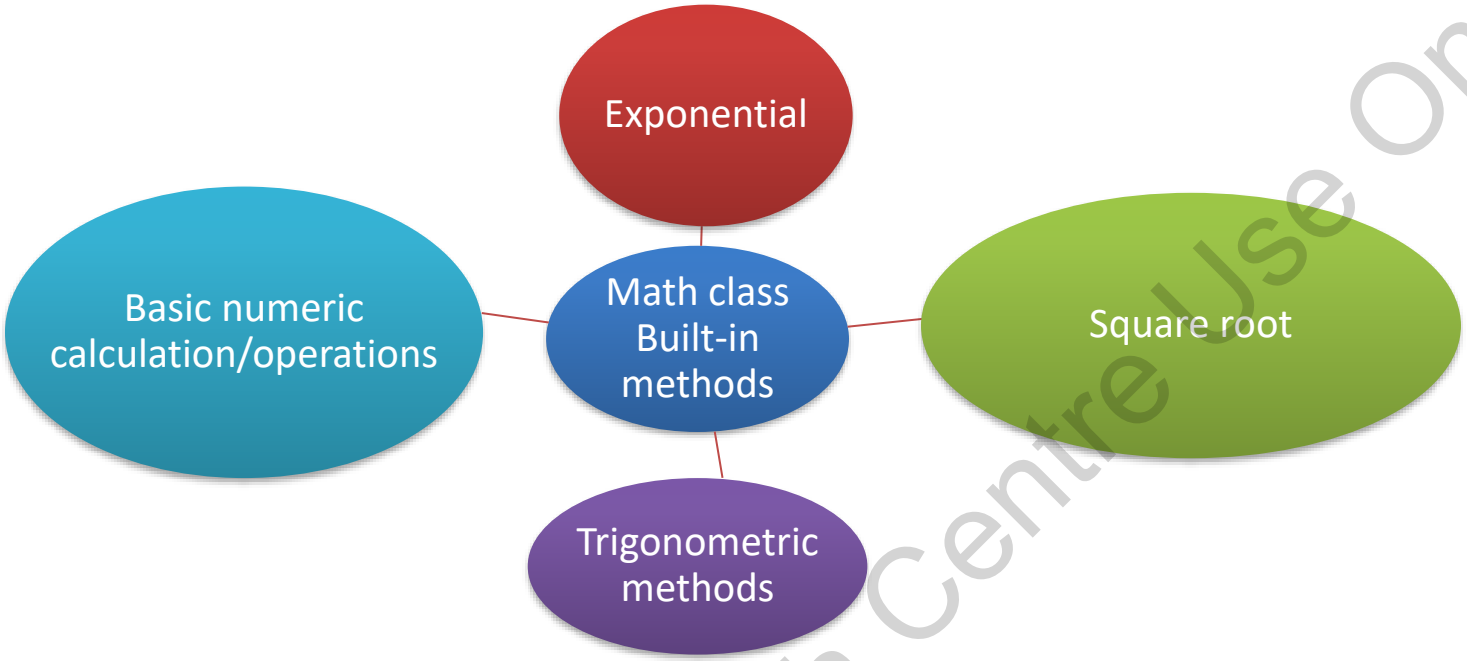
Removal of var as a valid class name

From Java 10 onwards you cannot use var as a class name

Removal of underscore as a valid identifier

From Java 9 onwards, you cannot use underscore as a valid identifier

Mathematical Methods in Java



They are located in the `Math` class in `java.lang` package

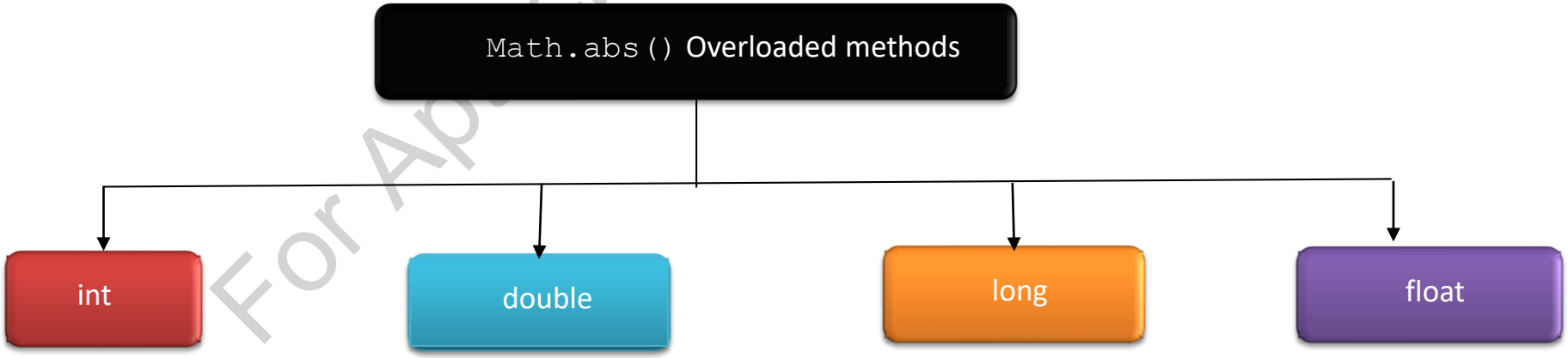
Advanced mathematical operations can also be performed with `Math` class

Basic Math Methods 1-3

Math.abs()

This method produces the absolute value of the given input in the calculation. `Math.abs()` only returns the positive value, even if the given input contains negative values

```
public class BasicMathDemo {
    public static void main(String args[]) {
        int abs1 = Math.abs(10); // abs1 = 10
        int abs2 = Math.abs(-20); // abs2 = 20
        System.out.println("Result A: "+abs1);
        System.out.println("Result B: "+abs2);
    }
}//displays result
```



Basic Math Methods 2-3

`Math.ceil()`

- This method rounds a floating-point value close to the integer value
- The rounded value is returned as a double

```
public class BasicMathDemo {
     public static void main(String args[]) {
          double objCeil=Math.ceil(6.454);
          System.out.println("Result as "+objCeil);
     }
}//displays result
```

After executing this code, `objCeil` contains the value 7.0 .

Basic Math Methods 3-3

Math.floor()

Method Name	Description	Example
Math.floorDiv()	This method is similar to floor method, but is combined with division. Math. floorDiv() divides an integer or long value by another one and rounds the resultant value to the closest integer value.	double output = Math. floorDiv(100,9); Output: 11.0
Math.min()	This method produces the smallest of the given values passed as inputs.	int newMin = Math.min(5, 12); Here, the newMin variable produces output as 5.
Math.max()	This method is similar to the Math.min() method with one difference that it produces the biggest value of the given inputs.	int newMax = Math. max(5, 12); Here, the newMax variable produces output as 12.
Math.round()	This method rounds a float or double to the closest integer. Math.round() method applies common mathematical rules for rounding the values.	double newDecrease = Math.round(44.324); double newIncrease = Math.round(44.654); Here, newDecrease variable produces output as 44.0 and the newIncrease variable produces 45 as output.
Math.random()	This method returns a random floating point value between 0 and 1 by default. However, Math.random can be applied to get a random number between 0 and n (any number within 100).	double newRand1 = Math.random(); double newRand2 = Math.random() * 50D; Output: newRand1: 0.7463562032119188 newRand2: 26.360465065790073

Exponential and Logarithmic Math Methods 1-5

Math class also contains a set of methods to perform exponential and logarithmic operations/calculations.

`Math.exp()`

This method produces e (Euler's number) increased to the power of the value given as a parameter.

```
public class ExpoandLogMathFunctons {
    public static void main (String args[]) {
        double newExpA = Math.exp(4);
        System.out.println("OutputA = " + newExpA);
        double newExpB = Math.exp(5);
        System.out.println("OutputB = " + newExpB);
    }
}//displays result
```

Here, newExpA and newExpB variables produce following output:

```
OutputA = 54.598150033144236
OutputB = 148.4131591025766
```

Exponential and Logarithmic Math Methods 2-5

Math.log()

Provides the logarithm of the given value. Math.log() method functioning under the basis of logarithm is i (Euler's number).

```
public class ExpoandLogMathFunctons {
    public static void main(String args[]) {
        double newLogA = Math.log(2);
        System.out.println("OutputA = " + newLogA);
        double newLogB = Math.log(100);
        System.out.println("OutputB = " + newLogB);
    }
}//displays result
```

Here, the newLogA and newLogB are the variables that produce following result:

```
OutputA = 0.6931471805599453
OutputB = 4.605170185988092
```

Exponential and Logarithmic Math Methods 3-5

`Math.log10()`

The only difference is it takes 10 as a base for calculating the logarithm instead of e (Euler's Number).

```
public class ExpoandLogMathFunctons {
      public static void main(String args[]) {
           double newLog10A=Math.log10(6);
           System.out.println("OutputA=" +newLog10A);
           double newLog10B=Math.log10(200);
           System.out.println("OutputB=" +newLog10B);
      }
}//displays result
```

Here, newLog10A and newLog10B are the variables that produce following result:

```
OutputA = 0.7781512503836436
OutputB = 2.3010299956639813
```

Exponential and Logarithmic Math Methods 4-5

Math.pow()

Takes two parameters and produces the value of the first parameter raised to the power of the second parameter.

```
public class ExpoandLogMathFunctions {
    public static void main(String args[]){
        double newPowerA = Math.pow(2, 4);
        System.out.println("OutputA as = " + newPowerA);
        double newPowerB = Math.pow(2, 5);
        System.out.println("OutputB as = " + newPowerB);
     }
}
```

Here, newPowerA and newPowerB produce following results:

```
OutputA as = 16.0
OutputB as = 32.0
```

Exponential and Logarithmic Math Methods 5-5

Math.sqrt()

It performs the square root operation for the given parameter.

```
public class ExpoandLogMathFunctions {
    public static void main(String args[]){
        double newSrootA=Math.sqrt(8);
        System.out.println("OutputA=" +newSrootA);
        double newSrootB=Math.sqrt(25);
        System.out.println("OutputB=" +newSrootB);
    }
}
```

Here, the newSrootA and newSrootB produce following results:

```
OutputA = 2.8284271247461903
OutputB = 5.0
```

Trigonometric Math Methods 1-4

Method	Description	Example

`Math.sin()`	Performs the sine operation. It calculates the sine value of the given angle value in radians.	`double newSin = Math.sin(Math.PI);`
		`System.out.println("The value of sin = " + newSin);`
		Output: `The value of sin = 1.2246467991473532E-16`

`Math.cos()`	Performs the cos operation. It calculates the cosine value of the given angle value in radians.	`double newTan = Math.tan(Math.PI);`
		`System.out.println("The value of tan = " + newTan);`
		Output: `The value of tan = 1.2246467991473532E-16`

`Math.asin()`	Performs the arc sine value calculation of a value between `1` and `-1`.	`double newAsin = Math.asin(Math.PI);`
		`System.out.println("The value of Asin = " + newAsin);`
		Output: `The value of Asin = NaN`

`Math.acos()`	Performs the arc cos value calculation of a value between `1` and `-1`.	`double newAcos = Math.acos(1.0);`
		`System.out.println("The value of acos = " + newAcos);`
		Output: `The value of acos = 0.0`

`Math.atan()`	Performs the arc tangent value calculation of a value between `1` and `-1`.	`double newAtan = Math.atan(1.0);`
		`System.out.println("The value of Atan = " + newAtan);`
		Output: `The value of Atan = 0.7853981633974483`

Trigonometric Math Methods 2-4

Method	Description	Example

`Math.sinh()`	Performs the hyperbolic sine value calculation of a given value between `1` and `-1`.	`double newSinh = Math.sinh(1.0);`
		`System.out.println("The value of sinh = " + newSinh);`
		Output: `The value of sinh = 1.1752011936438014`

`Math.cosh()`	Performs the hyperbolic cosine value calculation of a given value between `1` and `-1`.	`double newCosh = Math.cosh(1.0);`
		`System.out.println("The value of Cosh = " + newCosh);`
		Output: `The value of Cosh = 1.543080634815244`

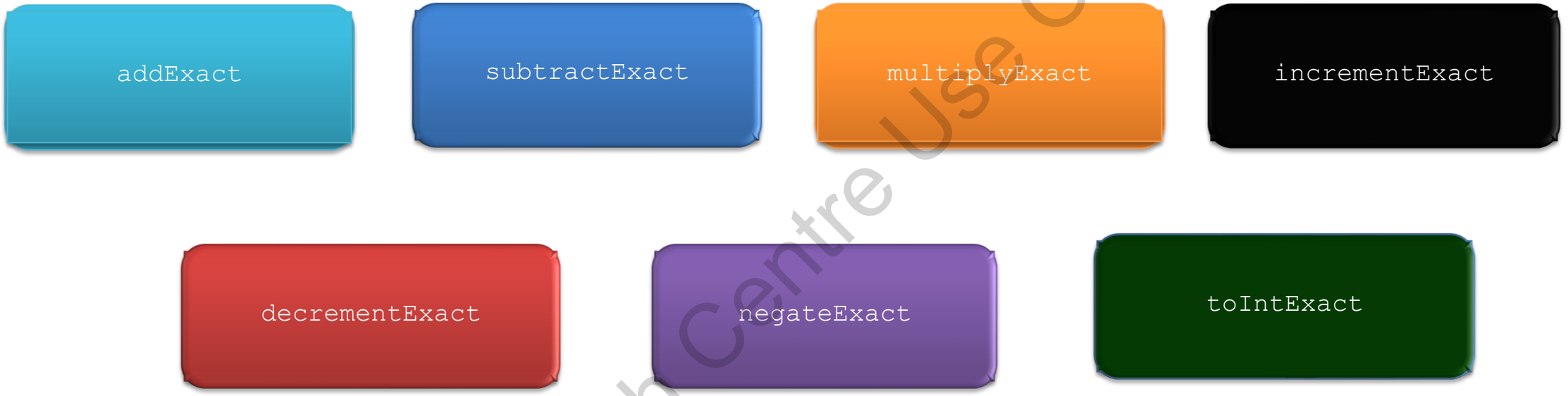
`Math.tanh()`	Performs the hyperbolic tangent value calculation of a given value between `1` and `-1`.	`double newTanh = Math.tanh(1.0);`
		`System.out.println("The value of tanh = " + newTanh);`
		Output: `The value of tanh = 0.7615941559557649`

`Math.toDegrees()`	Performs the convert operation of an angle in radians to degrees.	`double newDegrees = Math.toDegrees(Math.PI);`
		`System.out.println("Output = " + newDegrees);`
		Output = 180.0

`Math.toRadians()`	Performs an reverse operation of `Math.toDegrees()` method, it performs the convert operation of an angle in degrees to radians.	`double newRadians = Math.toRadians(180);`
		`System.out.println("Output = " + newRadians);`
		Output = 3.141592653589793

Trigonometric Math Methods 3-4

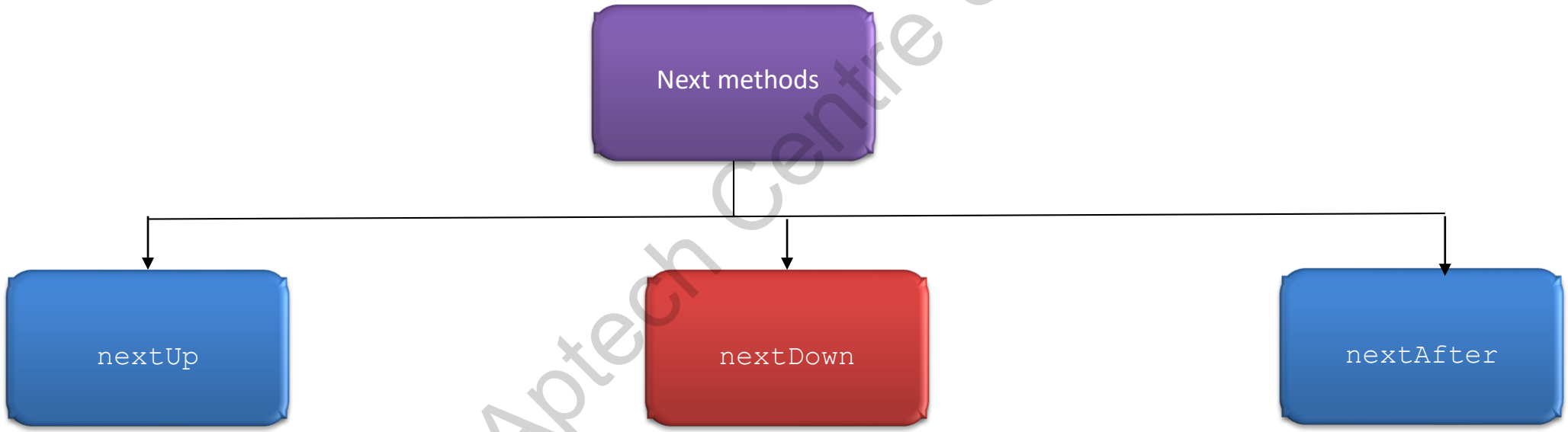
Exact Numeric Operations



Trigonometric Math Methods 4-4

Next Numeric Operations

Math class also introduces a set of methods to perform numeric operations that are applied where you must display the closest value of a given number.



Other Ways of Creating Objects 1-3

Using `newInstance`

To create an object of the class, you must use the `newInstance()` method of the class.

```
public class NewInstanceDemo{

  String name = "objNewInstanceDemo";

   public static void main(String[] args) throws ClassNotFoundException,
 InstantiationException, IllegalAccessException {

     Class cls = Class.forName("NewInstanceDemo");

     NewInstanceDemo obj = (NewInstanceDemo) cls.newInstance();

     System.out.println(obj.name);

 }

}
```

Other Ways of Creating Objects 2-3

Using `clone()` method:

To create a new object and copy all content of the previous object into it.

```
import java.util.*;
public class CloneDemo implements Cloneable {
   @Override
   protected Object clone() throws CloneNotSupportedException {
       return super.clone();
    }
    String name = "objCloneDemo";
    public static void main(String[] args) throws CloneNotSupportedException
    {
        CloneDemo obj1 = new CloneDemo();
        CloneDemo obj2 = (CloneDemo) obj1.clone();
        System.out.println(obj2.name);
    }
 }
```

Other Ways of Creating Objects 3-3

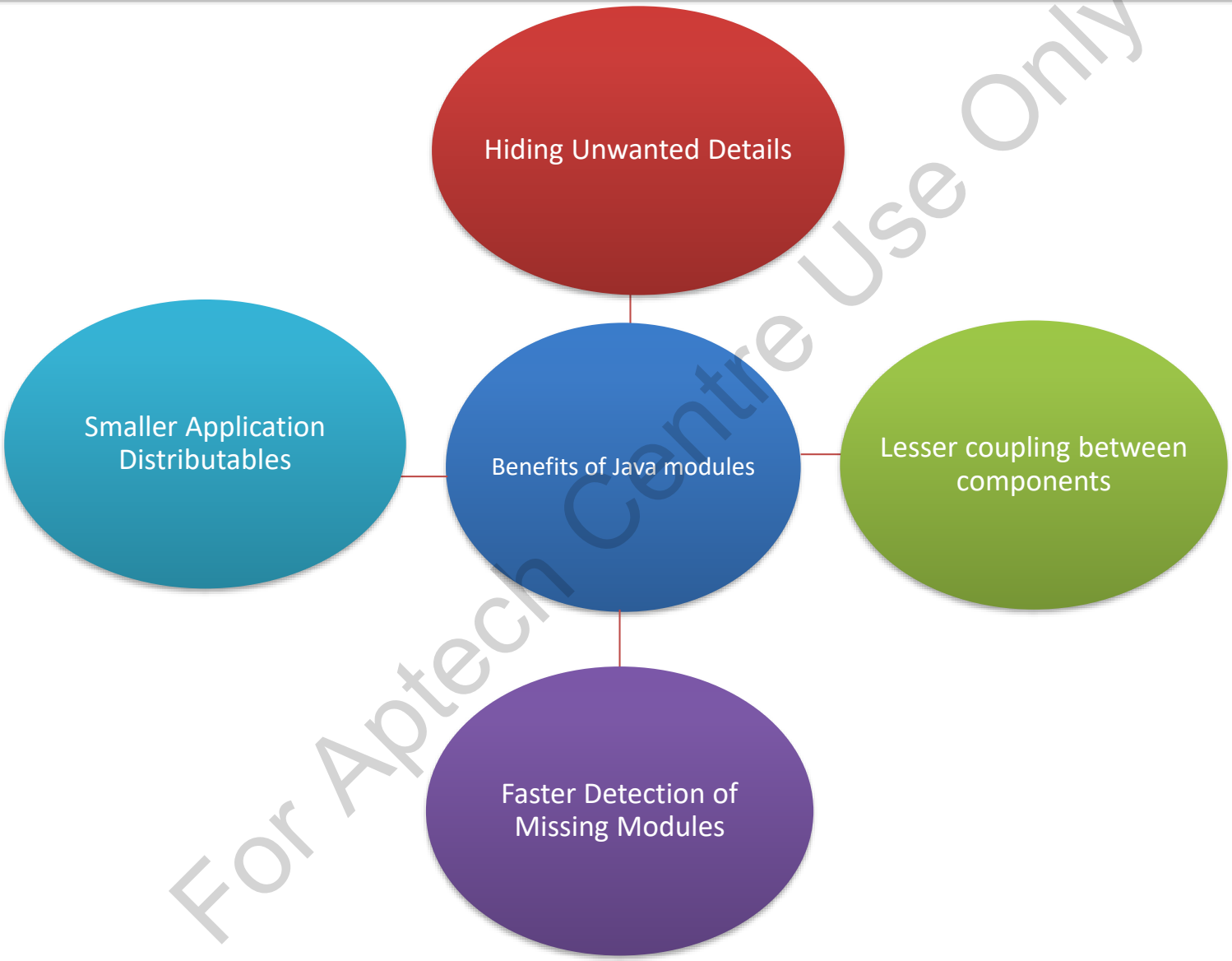
Using `newInstance()` method of Constructor class

Calls parameterized constructor and private constructor by using this newInstance() method.

```
import java.lang.reflect.*;
import java.util.*;
public class ReflectionDemo {
 private String name;
 ReflectionDemo()
    {
 }

    public void setName(String name) {
        this.name = name;
     }
   public static void main(String[] args) throws NoSuchMethodException,
                InstantiationException, IllegalAccessException,
                InvocationTargetException {
         Constructor<ReflectionDemo> constructor = ReflectionDemo.class.
            getDeclaredConstructor();
        ReflectionDemo objReflectionDemo = constructor.newInstance();
        objReflectionDemo.setName("objReflectionDemo");
        System.out.println(objReflectionDemo.name);
    }
 }
```

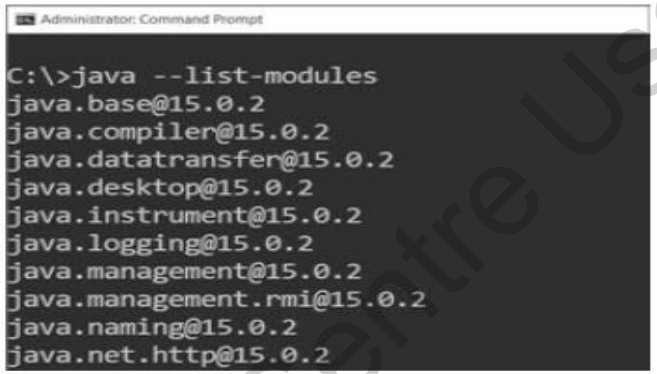
Java Modules



Types of Modules

System Modules:

These are Java SE and JDK system-defined modules. You can view a list of these by running the list-modules command.



Application Modules:

These modules are what developers want to build when they decide to use modules.

Automatic Modules:

Will have full read access to every other module loaded by the path.

Unnamed Module:

To maintain backward compatibility with previously-written Java code.

Creating a Java Module

- Create a module descriptor file
- Create the module declaration
- Add directives
- Set Up the Project
- Add Code to the Module
- Compile the Module
- Execute the module

Enhanced Switch Statement and Switch Expressions

- The traditional switch statement in earlier versions of Java has been enhanced in recent versions to include a new arrow syntax (also called 'switch labeled rules')

```
public class EnhancedSwitchStatementDemo{
  public static void main(String[] args) {
     System.out.println("Enhanced Switch Statement:");
     final int integer = 3;
     String numericString;
     switch (integer) {
           case 1 -> numericString = "one";
           case 2 -> numericString = "two";
           case 3 -> numericString = "three";
          default -> numericString = "N/A";
      }
      System.out.println("\t" + integer + " ==> " + numericString);
     }
}
```

Output:

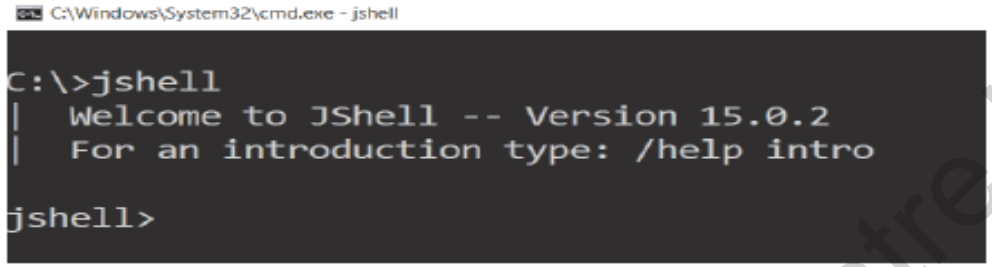
Enhanced Switch Statement:

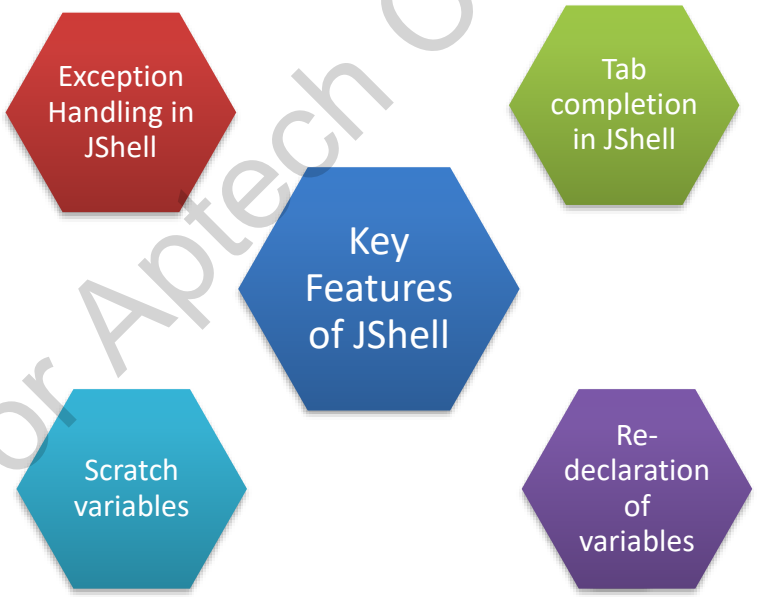
3 ==> three

JShell

- Java Shell tool (JShell) is an interactive tool that helps you try out Java code and easily explore options as you develop your program.

Start and Stop JShell





Hidden Classes and Sealed Classes

- Hidden classes are classes that cannot be used directly by bytecode of other classes
- Mainly intended for use by frameworks that generate classes at runtime and use them indirectly, via reflection
- The sealed scope modifier in Java 15 provides fine grained inheritance control for classes and interfaces
- Sealed classes are also useful for creating secure hierarchies by decoupling accessibility from extensibility

Summary

- Between Java 9 to 15, several features that existed in earlier versions have been deprecated or removed now and no longer recommended for use.
- Nashorn is a JavaScript engine that was used in Java versions 8 to 10 to compile JavaScript into Java bytecode. It was deprecated in Java 11.
- Advanced mathematical operations can be performed with new methods in Math class.
- A module can be defined as a group of closely related packages and resources along with a new module descriptor file.
- Recent Java versions have enhanced the switch statement and also provided a switch expression.
- JShell is an interactive tool that helps you try out Java code and easily explore options as you develop your program.
- In Java 15 onwards, you can define sealed classes and hidden classes.