Core java 1:

Given:  
   float result;  
   result=5/2;  
   System.out.println(result);  
  
What will be the output?

**1)**. 2

**2)**. **2.0**

**3)**. 2.5

**4)**. error

**Solution** :  
option [2] is correct

**Attempted** :  
option [2] is attempted

What is true about the following code?          
  
1. enum EnumDemo { A }  
2. class Test {  
3. enum EnumD { B }  
4. void my\_method() {  
5. enum EnumC { D }  
6. } }

**1)**. The code compiles without any error

**2)**. The code compiles if we remove line number 1

**3)**. **The code compiles if we remove line number 5**

**4)**. None of the above is correct

**Solution** :  
option [3] is correct

**Attempted** :  
option [3] is attempted

public class test {           
                              
    static void methodA(short s) {  
     System.out.println("methodA(short) called");  
    }  
  
    static void methodA(int i) {  
     System.out.println("methodA(int) called");  
    }  
                           
                         
    static void methodB(float f) {  
    System.out.println("methodB(float) called");  
    }  
  
    static void methodB(double d) {  
    System.out.println("methodB(double) called");  
    }  
                           
    public static void main(String args[]) {  
                             
    methodA(5);                                                   
    methodB(5.2);  
  
    }  
}

**1)**. methodA(short) called

**2)**. **methodA(int) called**

**3)**. Exception

**4)**. Compilation Error

**Solution** :  
option [2] is correct

**Attempted** :  
option [2] is attempted

When access modifier is omitted from the definition of the member of a class. The member has?

**1)**. **default access**

**2)**. public access

**3)**. private access

**4)**. protected access

**Solution** :  
option [1] is correct

**Attempted** :  
option [1] is attempted

 Which of the following are features of Java Programming Language?

**1)**. Robust

**2)**. Multithreaded

**3)**. Interpreted

**4)**. **All of the above**

**Solution** :  
option [4] is correct

**Attempted** :  
option [4] is attempted

Which of the following statements are true?

**1)**. No arg constructor is allways supplied by the Compiler

**2)**. **Constructors cannot be overloaded**

**3)**. **Constructors cannot have return type.**

**4)**. Constructors can be static

**Solution** :  
option [3] is correct

**Attempted** :  
option [2] is attempted

Memory deallocation in java is done by?

**1)**. Programmer

**2)**. Operating system

**3)**. **Garbage collector**

**4)**. None of the above

**Solution** :  
option [3] is correct

**Attempted** :  
option [3] is attempted

What all gets printed when the following code is compiled and run?   
Select the correct answers:  
  
public class Test {  
   public static void main(String args[])   
{  
      for(int i = 0; i < 2; i++)   
{  
         for(int j = 2; j>= 0; j--)   
{  
            if(i == j) break;  
           System.out.println("i=" + i + " j="+j);  
         }  
      }  
   }  
}   
  
1. i=0 j=0   
2. i=0 j=1   
3. i=0 j=2   
4. i=1 j=0   
5. i=1 j=1   
6. i=1 j=2

**1)**. 1,3,5

**2)**. 2,4,6

**3)**. 1,2,5

**4)**. **3,2,6**

**Solution** :  
option [4] is correct

**Attempted** :  
option [4] is attempted

Given:  
1. public class Demo {  
2. public static void main (String[] args) {  
3. byte var1 = 127;  
4. byte var2 = 126;  
5. byte result = var1 + var2;  
6. }  
7. }  
  
Which statement is true?

**1)**. Compilation succeeds and d takes the value 253.

**2)**. **Line 5 contains an error that prevents compilation.**

**3)**. Line 5 throws an exception indicating "Out of range"

**4)**. Line 3 and 4 contain errors that prevent compilation.

**Solution** :  
option [2] is correct

**Attempted** :  
option [2] is attempted

Given the following code fragment:  
  
 XXXX  choice ; // variable choice is declared and initialized here  
 switch( choice ) {  
   case 100 : System.out.println("One hundred");break ;  
   case 20 : System.out.println("Twenty");break ;  
   case 30 : System.out.println("Thirty");break ;  
 }  
  
Choose the declarations of choice which will not cause a compiler error.

**1)**. byte choice = 100 ;

**2)**. short choice = 100 ;

**3)**. int choice = 300 ;

**4)**. **All of the above**

**Solution** :  
option [4] is correct

**Attempted** :  
option [4] is attempted

Which of the following are Object class methods?

**1)**. toString()

**2)**. equals()

**3)**. hashCode()

**4)**. **All of the above**

**Solution** :  
option [4] is correct

**Attempted** :  
option [4] is attempted

The Scanner class is found in  \_\_\_\_\_\_ package.

**1)**. java.lang

**2)**. **java.util**

**3)**. java.io

**4)**. None of the above

**Solution** :  
option [2] is correct

**Attempted** :  
option [2] is attempted

Integer var1 = new Integer(2);   
Integer var2 = new Integer(2);   
  
What happens when you do if (var1==var2)?

**1)**. 'TRUE

**2)**. **'FALSE**

**3)**. compilation error

**4)**. runtime Exception

**Solution** :  
option [2] is correct

**Attempted** :  
option [2] is attempted

What will be the output of the following code snippet?         '  
  
String str1 = new String( "hello" );  
String str2 = "hello";  
         if  (str1==str2)   
           System.out.println( "Equals");  
        else  
           System.out.println( "Not Equal");

**1)**. Equal

**2)**. **Not Equal**

**3)**. Compilation error

**4)**. None of the above

**Solution** :  
option [2] is correct

**Attempted** :  
option [2] is attempted

class Shape  
{  
final public double calArea(){}  
}  
public class Circle extends Shape  
{  
int radius;  
public Circle(int radius){  
this.radius = radius;  
}  
public double calArea(){  
return 3.142\*radius\*radius;  
}  
public static void main(String []args){  
Shape obj = new Circle(5);  
System.out.println(obj.calArea());  
}  
}What will be the output of above code?

**1)**. It will display area of circle with radius 5

**2)**. no output

**3)**. **compilation error**

**4)**. runtime Exception

**Solution** :  
option [3] is correct

**Attempted** :  
option [3] is attempted

 What will happen when you attempt to compile and run the following class?        
  
class Base{  
Base(int var)  
{  
    System.out.println("Base");  
}  
}  
class Test extends Base{  
public static void main(String argv[]){  
        Test obj = new Test();  
}  
}

**1)**.  Compiles without any problem

**2)**.  Compiles and produces output "Base"

**3)**. **Generates Compile time error**

**4)**. None of the above

**Solution** :  
option [3] is correct

**Attempted** :  
option [3] is attempted

class Employee  
{  
String name;  
int id;  
public Employee(String name,int id)  
{  
this.name = name;  
this.id=id;  
}  
}  
public class Manager extends Employee  
{  
public static void main(String []args)  
{  
Manager mgr = new Manager();  
}  
}  
What will happen after execution of above code?

**1)**. Manager class object will be successfully created

**2)**. **compile time error as there is no default constructor in class Employee**

**3)**. Manager object will be successfully created after implementing default constructor in Manager class

**4)**. Default constructor need to be implemented in both classes for successful creation of Manager class object

**Solution** :  
option [2] is correct

**Attempted** :  
option [2] is attempted

Which of the given syntax is correct to display  date-time with the time zone in the ISO-8601 calendar system, such as 2007-12-03T10:15:30+01:00 Europe/Paris.   
Assume the zone is Asia/kolkata.

**1)**. **ZonedDateTime objt = ZonedDateTime.of(LocalDateTime.now(), ZoneId.of("Asia/Kolkata"));**

**2)**. LocalDate date = LocalDate.now(ZoneId.of("Asia/Kolkata"));

**3)**. ZonedDateTime time = new ZonedDateTime("Asia/Kolkata");

**4)**. ZonedDateTime time = new ZonedDateTime.of("Asia/Kolkata");

**Solution** :  
option [1] is correct

**Attempted** :  
option [1] is attempted

Which of the given LocalDate class method is used to obtain current date?

**1)**. **now()**

**2)**. parse()

**3)**. format()

**4)**. newInstance()

**Solution** :  
option [1] is correct

**Attempted** :  
option [1] is attempted

 class Demo  
{}  
public class SubDemo extends Demo  
{  
public static void main(String []args){  
Demo obj = new SubDemo();  
System.out.println(obj instanceof Demo);}  
}What will be the output of above code?

**1)**. FALSE

**2)**. **TRUE**

**3)**. Demo

**4)**. SubDemo

**Solution** :  
option [2] is correct

**Attempted** :  
option [2] is attempted

 Which of the following statements are true related to interface ?

**1)**. **Interface doesn't alllow to create object .**

**2)**. **Multiple inheritence can be possible in interface.**

**3)**. Class can implement only one interface .

**4)**. Interface can't allow to declare a member variable.

**Solution** :  
option [1,2] are correct

**Attempted** :  
option [1,2] are attempted

What is the Regular Expression Matching Zero or More Specific Characters?

**1)**. \

**2)**. $

**3)**. **\***

**4)**. ^

**Solution** :  
option [3] is correct

**Attempted** :  
option [3] is attempted

 Consider the following code:   
  
interface MyInterface {  
  
// Method declaration code  
  
 }   
Which of the following code snippet shows the wrong way to declare Method in interface ?

**1)**. public abstract boolean isValid();

**2)**. public boolean isValid();

**3)**. **protected boolean isValid();**

**4)**.  boolean isValid();

**Solution** :  
option [3] is correct

**Attempted** :  
option [3] is attempted

On which of the given options abstract modifier can be used?

**1)**. constructor

**2)**. static method

**3)**. **non-static methods**

**4)**. **class**

**Solution** :  
option [3,4] are correct

**Attempted** :  
option [3,4] are attempted

Which of the given syntax is correct for interface implementation in Java 8?

**1)**. **interface Demo{  
}**

**2)**. **interface Demo{  
default int getNumber(){return 0;}  
}**

**3)**. **interface Demo{  
static int getNumber(){return 0;}  
}**

**4)**. interface Demo  
{  
int getNumber(){return 0;}  
}

**Solution** :  
option [1,2,3] are correct

**Attempted** :  
option [1,2,3] are attempted

 class Shape  
{  
Shape(){}  
}  
public class Triangle extends Shape  
{  
int base,height;  
public Triangle(int base,int height){  
this.base = base;  
this.height=height;  
}  
public double calArea(){  
return o.5\*base\*height;  
}  
public static void main(String []args){  
Shape ref = new Triangle(3,4);  
System.out.println(ref.calArea());  
}Which change need to be done in the Shape class for successful execution of program?

**1)**. Declare Shape as abstract class

**2)**. Remove default constructor from Shape class

**3)**. declare abstract double calArea() in Shape class

**4)**. **declare abstract double calArea() in Shape class and declare Shape class as abstract class**

**Solution** :  
option [4] is correct

**Attempted** :  
option [4] is attempted

 public class RegexMatches  
{  
    private static String regex = "dog";  
    private static String input = "The dog says meow. " +"All dogs say meow.";  
    private static String replace = "cat";  
  
    public static void main(String[] args) {  
        //line no.1  
         
       Matcher m = p.matcher(input);   
       INPUT = m.replaceAll(replace);  
       System.out.println(input);  
   }  
}Which code need to be inserted at line no.1 to execute it successfully?

**1)**. **Pattern p = Pattern.compile(regex);**

**2)**. Pattern p = Pattern.test(regex);

**3)**. Pattern p = Pattern.matches(regex);

**4)**. Pattern p = new Pattern(regex);

**Solution** :  
option [1] is correct

**Attempted** :  
option [1] is attempted

 public class Demos {  
        
      public static void main(String[] args) {  
       String input = "Hello Welcome";  
       String pattern = "\\sHello\\sWelcome\\s";  
       boolean flag = Pattern.matches(pattern, input);  
       System.out.println(flag);  
      }  
}

**1)**. TRUE

**2)**. **FALSE**

**3)**. compilation error

**4)**. exception

**Solution** :  
option [2] is correct

**Attempted** :  
option [2] is attempted

Consider the following code:   
  
interface Greek { }   
  
class Alpha implements Greek { }   
  
class Beta extends Alpha {}   
  
class Delta extends Beta   
 {   
    public static void main( String[] args )   
     {   
       Beta obj = new Beta(); // insert code here   
     }  
 }  
  
Which of the following code snippet when inserted individual at the commented line   
(// insert code here), will cause a  java.lang.ClassCastException?

**1)**. Greek objGrk = (Beta)(Alpha)obj;

**2)**. Alpha objAlpha = obj;

**3)**. Greek objGrk = (Alpha)obj;

**4)**.  Beta objBeta = (Beta)(Alpha)obj;

**5)**. **Greek objGrk = (Delta)obj;**

**Solution** :  
option [5] is correct

**Attempted** :  
option [5] is attempted

 Which of the following is FALSE about abstract classes in Java?

**1)**. If we derive an abstract class and do not implement all the abstract methods, then the derived class should also be marked as abstract using 'abstract' keyword

**2)**. Abstract classes can have constructors

**3)**. A class can be made abstract without any abstract method

**4)**. **A class can inherit from multiple abstract classes**

**Solution** :  
option [4] is correct

**Attempted** :  
option [4] is attempted

What is the output of the following program?  
  
class Test {  
public static void main(String[] args) {  
try {  
doMath(5);  
System.out.print("hi");  
}  
finally { System.out.println(" from finally"); }  
}  
public static void doMath(int den) {  
int num = 7 / den;  
}  
}

**1)**. **hi from finally**

**2)**. hi  
from finally

**3)**. prints hi from finally 2 times

**4)**. None of the above

**Solution** :  
option [1] is correct

**Attempted** :  
option [1] is attempted

class ArrayDemo  
{  
public static void main(String []args){  
String str = "Hello World";  
int []arr = {1,2,3,4,5};  
display(arr,str);  
}  
public static void display(int …arr,String str)  
{  
for(int num:arr){System.out.println(num);}  
System.out.println(str);  
}  
}

**1)**. 12345Hello World

**2)**. Hello World

**3)**. NumberFormatException

**4)**. **Compilation Error**

**Solution** :  
option [4] is correct

**Attempted** :  
option [4] is attempted

 public class Demos {  
        public static void main(String[] args) {  
       int arr[5]={1,2,3,4,5};  
        for(int ele:arr){  
        System.out.print(str);}  
}  
What will be the output of above code?

**1)**. 12345

**2)**. no output

**3)**. ArrayIndexOutOfBounds Exception

**4)**. **Compilation Error**

**Solution** :  
option [4] is correct

**Attempted** :  
option [4] is attempted

**Q.**\_\_\_\_\_\_\_\_\_\_is raised if I do not provide the String array as the argument to the main method.

**1)**. NullPointerException

**2)**. IllegalAccessException

**3)**. **NoSuchMethodError**

**4)**. **None of the above**

**Solution** :  
option [3] is correct

**Attempted** :  
option [4] is attempted

What will be the output of following program ?  
public class Foo   
{    
    public static void main(String[] args)   
    {  
        try   
        {   
            return;   
        }   
        finally   
        {  
            System.out.println( "Finally" );   
        }   
    }   
}

**1)**. **Finally**

**2)**. Compilatation fails .

**3)**. No output.

**4)**. runtime Exception

**Solution** :  
option [1] is correct

**Attempted** :  
option [1] is attempted

In Java arrays are \_\_\_\_\_\_\_\_\_\_\_.

**1)**. **objects**

**2)**. object references

**3)**. primitive data type

**4)**. All of the above

**Solution** :  
option [1] is correct

**Attempted** :  
option [1] is attempted

Which of the following statements are true related to exception handling in java ?

**1)**. UserDefined exception can be created by extending from RunTimeException class

**2)**. Throwable is the base class of Error and Exception class .

**3)**. Checked Exception need to be handled either by try and catch block or by using throws keyword in a code before compilation.

**4)**. **All of the above**

**Solution** :  
option [4] is correct

**Attempted** :  
option [4] is attempted

public class Test{  
        public static void main(String[] args){  
                int[] a = new int[0];  
                System.out.print(a.length);  
        }  
}

**1)**. 1

**2)**. **0**

**3)**. Compilation error, arrays cannot be initialized to zero size.

**4)**. Compilation error, it is a.length() not a.length

**Solution** :  
option [2] is correct

**Attempted** :  
option [2] is attempted

 public class Test{  
        public static void main(String[] args){  
                int[] a = new int[0];  
                System.out.print(a.length);  
        }  
}

**1)**. 1

**2)**. **0**

**3)**. Compilation error, arrays cannot be initialized to zero size.

**4)**. Compilation error, it is a.length() not a.length

**Solution** :  
option [2] is correct

**Attempted** :  
option [2] is attempted