|  |  |  |
| --- | --- | --- |
| **Question Name (Optional)** | **The Question Itself** | **Correct answer** |
| CoreJava- AccessSpecifiers-001 | Consider the following listed items:   1. Employee() { } 2. public Employee() { } 3. private Employee() { }   Consider the following statements:   1. no-argument constructor, that does not allow instantiation from within the package 2. no-argument constructor, that does not allow instantiation from outside the package 3. no-argument constructor   Which of the following option gives the exact matches of above listed items and statements? | A-II, B-III, C-I |
| CoreJava- AccessSpecifiers-002 | Which of the following statement is true regarding constructors? | Default Constructors are optional only for the classes that does not have constructors |
| CoreJava- AccessSpecifiers-003 | Which of the following statement is true regarding parameterized constructors? | Parameterized constructors can accept variable arguments as their parameters |
| CoreJava- AccessSpecifiers-004 | Consider the following code:  class Student { private String name;  public Student() { }  public Student(String name) { this.name = name;  this();  }  }  Which of the following statement is true regarding the above code? | The this() call should be the first statement in the constructor |

|  |  |  |
| --- | --- | --- |
| CoreJava- AccessSpecifiers-005 | Consider the following listed items:   1. a method declared as final 2. a method declared as abstract 3. a method declared as private   Consider the following statements:   1. Will not be available in sub classes 2. Will deny overriding the method 3. Will not allow instantiating the class   Which of the following option gives the exact matches of above listed items and statements? | A-II, B-III, C-I |
| CoreJava- AccessSpecifiers-006 | Consider the following code:  public class TestVarArgs {  public static void main(String args[]) { Employee e = new Employee();  }  static int add(int a, int b) { return a + b;  }  static int add(int a, int b, int c) { return add(a, b) + c;  }  static int add(int a, int b, int c, int d) { return add(a, b ,c) + d;  }  }  Which of the following statement gives the alternative way of implemeting the above add() methods? | Using Var-Args |

|  |  |  |
| --- | --- | --- |
| CoreJava- AccessSpecifiers-007 | Which of the following statements are true regarding methods?   1. A method name can start with '$' symbol 2. A method that returns a value can have java.lang.Object as its return type irrespective of the type of the value it returns 3. A member method cannot have its owner class as its return type 4. A method that has void as its return type cannot have a return statement 5. A method cannot have multiple return statements | 1,2 |
| CoreJava- AccessSpecifiers-008 | Which of the following statements are true regarding var- args?   1. The main method arguments can be declared as var- args of String 2. The var-arg variable can be iterated using enhanced for loop 3. Normal arguments and var-args cannot be mixed 4. Method declarations in interface cannot have var-arg type parameters 5. Var-arg can be a return type for a method | 1,2 |
| CoreJava- AccessSpecifiers-009 | Consider the following listed items:   1. Differing by Signatures 2. Code that executes before main() method 3. Code that executes before constructor   Consider the following statements:   1. Instance Block 2. Method Overloading 3. Static Block   Which of the following option gives the exact matches of above listed items and statements? | A-II, B-III, C-I |
| CoreJava- AccessSpecifiers-010 | Which of the following statement is true regarding method overloading? | Methods can be overloaded across inherited classes |

|  |  |  |
| --- | --- | --- |
| CoreJava- AccessSpecifiers-011 | Which of the following statements are true regarding static block and instance block?   1. Static methods can invoke static blocks 2. Static blocks can call static methods 3. Instance block can invoke current object's constructor 4. Constructor can invoke instance blocks 5. Static blocks are executed only once in the class's life cycle | 2,5 |
| CoreJava- AccessSpecifiers-012 | Consider the following code:  public class TestOverloading { int \_length(String s) {  return s.length();  }  float \_length(String s) { return (float) s.length();  }  }  Which of the following statement is true regarding the above code? | Both the length() methods are duplicated methods |
| CoreJava- AccessSpecifiers-013 | Consider the following listed items:   1. public classes 2. private classes 3. abstract classes   Consider the following statements:   1. Cannot be used without inheriting 2. Not possible in Java 3. Can be imported from other packages   Which of the following option gives the exact matches of above listed items and statements? | A-III, B-II, C-I |

|  |  |  |
| --- | --- | --- |
| CoreJava- AccessSpecifiers-014 | Which of the following statements are true?   1. Static modifier is applicable only for inner classes 2. Final modifier is not applicable for abstract classes 3. Package level members in a public class will be available to the inherited class outside the package. 4. Private members of a class are not accessible through its object instance 5. Protected members of a class are accessible through its object instance | 1,2,5 |
| CoreJava- AccessSpecifiers-015 | Which of the following statement is true regarding access specifiers? | Protected level access is only for members, not for classes |
| CoreJava- AccessSpecifiers-016 | Consider the following code:  class One {  private void method() { System.out.println("method in One");  }  }  class Two extends One { void method() {  System.out.println("method in Two");  }  }  Which of the following statements are true regarding the above code?   1. method() is a overloaded method 2. method() is a overridden method 3. They are independent methods, no connection between them 4. method() in class Two is polymorphic 5. the access level of method() cannot be expanded from private to package level | 3,4 |

|  |  |  |
| --- | --- | --- |
| CoreJava- AccessSpecifiers-017 | Consider the following code:  public class Normal2VarArgsTester { public static void main(String args[]) {  System.out.println(stringProcessor("Hi", "Hello", "Welcome", "Bye"));  }  static String stringProcessor(String s1, String s2, String s3, String s4) {  System.out.println("Normal Version"); return s1 + s2 + s3 + s4;  }  }  The above code contains a method stringProcessor(), that does some string processing on 4 string objects passed to it.  The same is tested by calling it from main method(). The output generated by the above code is  Normal Version HiHelloWelcomeBye  Now, it is decided to add an overloaded version of the same stringProcessor() method with Var-args that accepts String as its parameter. | Normal Version HiHelloWelcomeBye Var-arg Version  HiHelloWelcomeByeGood Night |

|  |  |  |
| --- | --- | --- |
| CoreJava- AccessSpecifiers-018 | Consider the following code:  public class Normal2ArrayTester { public static void main(String args[]) {  System.out.println(numericProcessor(10, 20, 30, 40));  }  static Integer numericProcessor(Integer n1, Integer n2, Integer n3, Integer n4) {  System.out.println("Normal Version"); return n1 + n2 + n3 + n4;  }  }  The above code contains a method numericProcessor(), that does some numeric processing on 4 Integer objects passed to it.  The same is tested by calling it from main method(). The output generated by the above code is  Normal Version 100  Now, it is decided to add an overloaded version of the same numericProcessor() method with Integer array as its parameter.  The following code is added to the above class. | Shows a compile time error for the line that is newly added to the main() method for testing the new code |

|  |  |  |
| --- | --- | --- |
| CoreJava- AccessSpecifiers-019 | Consider the following code:  public class Arrays2VarArgsTester { public static void main(String args[]) {  System.out.println(numericProcessor(new Float[]  {1.0f, 2.0f, 3.0f, 4.0f}));  }  static Integer numericProcessor(Integer n[]) { System.out.println("Arrays Version"); Integer r = 0;  for(Integer i:n) { r += i;  }  return r;  }  }  The above code contains a method numericProcessor(), that does some numeric processing on 4 Integer objects passed to it.  The same is tested by calling it from main method(). The output generated by the above code is  Normal Version 100 | Shows a compile time error for the line that is newly added to the main() method for testing the new code |

|  |  |  |
| --- | --- | --- |
| CoreJava- AccessSpecifiers-020 | Consider the following code:  class MethodProvider { public String message; void method() {  System.out.println("Built-in method:" + message);  }  }  public class AnonymousOverloading { public static void main(String args[]) {  MethodProvider mp = new MethodProvider() { void method(String message) {  this.message = message; System.out.println("Built-out method:" +  message);  }  };  mp.method("Hello"); mp.method();  }  }  Which of the following option gives the output generated by the above code ? | Compile time error 'The method method() in the type MethodProvider is not applicable for the arguments (String)' |

|  |  |  |
| --- | --- | --- |
| CoreJava- AccessSpecifiers-021 | Consider the following code:  public class TestVarArgs {  public static void main(String args[]) { System.out.println(elementProcessor("A", "B", "C")); System.out.println(elementProcessor(1, 2, 3)); System.out.println(elementProcessor(10.0f, 20.0f,  30.0f));  }  static String elementProcessor(/\* CODE 1 \*/  ...elements) {  String result = "";  for(/\* CODE 2 \*/ element:elements) { result += element.toString();  }  return result;  }  }  Which of the following code snippet when replaced to those comments /\* CODE 1 \*/ and /\* CODE 2 \*/ in the above code will generate the following output?  ABC 123  10.020.030.0 | CODE 1 - Object CODE 2 - Object |

|  |  |  |
| --- | --- | --- |
| CoreJava- AccessSpecifiers-022 | Consider the following code:  class Student { private Integer id; private String name;  private Double salary;  public Student() { }  public Student(/\* CODE \*/) { for(int i=0; i<p.length; i++){  switch(i) {  case 0: id = (Integer) p[i]; break; case 1: name = (String) p[i]; break; case 2: salary = (Double) p[i]; break;  }  }  }  @Override  public String toString() { return "Id:" + id + "\n" + "Name:" + name + "\n" +  "Salary:" + salary;  }  }  public class VarArgConstructor { | Object …o |

|  |  |  |
| --- | --- | --- |
| CoreJava- AccessSpecifiers-023 | Consider the following code:  class Student { private Integer id; private String name;  private Double salary;  public Student() { }  public Student(Integer id, String name, Double salary) { this.id = id;  this.name = name; this.salary = salary;  }  public Student(Student s) {  /\* CODE \*/  }  }  public class TestCopyConstructor { public static void main(String args[]) {  Student s1 = new Student(100, "Mr.X", 28000.0); Student s2 = new Student(s1);  }  }  Which of the following code snippet when replaced for the comment /\* CODE \*/ in the above program, will | this(s.id, s.name, s.salary); |

|  |  |  |
| --- | --- | --- |
| CoreJava- AccessSpecifiers-024 | Consider the following code:  class Student { private Integer id; private String name;  private Double salary;  public Student() { }  public Student(Integer id, String name, Double salary) { this.id = id;  this.name = name; this.salary = salary;  }  public void setStudentDetails(Student s) { System.out.println("Setting Student Details"); this(s.id, s.name, s.salary);  }  public String getStudentDetails() { String result = "";  result += "Id:" + id + "\n" + "Name:" + name + "\n" + "Salary:" + salary;  return result;  }  }  public class TestMethodCall { | this() method cannot be called from normal methods |

|  |  |  |
| --- | --- | --- |
| CoreJava- AccessSpecifiers-025 | Consider the following code:  class SuperClass { protected int param1; protected int param2;  }  class SubClass extends SuperClass { private int param3;  private int param4;  SubClass() { }  SubClass(int param1, int param2, int param3, int param4) {  /\* CODE \*/ this.param3 = param3; this.param4 = param4;  }  }  public class InitParams {  public static void main(String args[]) { SubClass sub = new SubClass(1, 2, 3, 4);  }  }  Which of the following code snippets when replaced to | 4,5 |

|  |  |  |
| --- | --- | --- |
| CoreJava- AccessSpecifiers-026 | Consider the following code:  class Person { private Integer id;  private String name;  }  class Employee extends Person { private Double salary;  Employee() { }  Employee(Integer id, String name, Double salary) { this.id = id;  this.name = name; this.salary = salary;  }  @Override  public String toString() { return "Id:" + this.id + "\n" +  "Name:" + this.name + "\n" + "Salary:" + this.salary;  }  }  public class TestEmployee {  public static void main(String args[]) { | Compile time error |

|  |  |  |
| --- | --- | --- |
| CoreJava- AccessSpecifiers-027 | Consider the following code:  class SuperClass {  Number giveCalculated(Integer a, Integer b) { System.out.println("SuperClass");  return a+b \* a-b;  }  }  class SubClass extends SuperClass {  /\* CODE \*/  }  Which of the following codes when exclusively replaced to the comment /\* CODE \*/ in the above code will make the above program to compile properly?   1. Float giveCalculated(Float a, Float b) { return a+b \* a-b;   }   1. Integer giveCalculated(Integer a, Integer b) { return a+b \* a-b;   }   1. Number giveCalculated(Integer a, Integer b) { return a+b \* a-b; | 1,4,5 |

|  |  |  |
| --- | --- | --- |
| CoreJava- AccessSpecifiers-028 | Consider the following code: package me;  import me.you.YourSharedProperty;  /\* CODE 1 \*/ class MyProperty { void method() {  YourSharedProperty ysp = new YourSharedProperty();  }  }  /\* CODE 2 \*/ class MySharedProperty { void method() {  MyProperty mp = new MyProperty();  }  }  package me.you;  import me.MySharedProperty;  /\* CODE 3 \*/ class YourProperty { void method() {  MySharedProperty msp = new MySharedProperty();  }  } | 1,2 |

|  |  |  |
| --- | --- | --- |
| CoreJava- AccessSpecifiers-029 | Consider the following code:  class SuperClass {  public /\* CODE 1 \*/ void method1() {  /\* CODE 3 \*/  }  }  class SubClass extends SuperClass { public /\* CODE 2 \*/ void method2() {  /\* CODE 4 \*/  }  }  Which of the following code snippets when exclusively replaced to the comments /\* CODE 1 \*/ /\* CODE 2 \*/ /\* CODE 3 \*/ and /\* CODE 4 \*/ in the above code, will make the code compile properly?   1. CODE 1 - static CODE 2 - static   CODE 3 - new SubClass().method2(); CODE 4 - new SuperClass().method1();   1. CODE 1 - No Code CODE 2 - No Code   CODE 3 - new SubClass().method2(); CODE 4 - new SuperClass().method1(); | 1,2,4 |

|  |  |  |
| --- | --- | --- |
| CoreJava- AccessSpecifiers-030 | Consider the following code: package com.teststatic;  class Static { static int i=10; static {  try {  Class.forName("com.teststatic.Static"); i++;  } catch(ClassNotFoundException cnfe) { System.out.println("Class Not Found");  }  }  }  public class TestStatic {  public static void main(String args[]) { try {  Class.forName("com.teststatic.Static");  }catch(ClassNotFoundException cnfe) { System.out.println("Class Not Found");  }  System.out.println(Static.i);  }  } | Class Not Found 10 |

|  |  |  |
| --- | --- | --- |
| CoreJava- AccessSpecifiers-031 | Consider the following code: package com.testinstance;  class Instance { static int i=100;  {  try {  Class.forName("com.testinstance.Instance"); i++;  } catch(ClassNotFoundException cnfe) { System.out.println("Class Not Found");  }  }  }  public class TestInstance {  public static void main(String args[]) { try {  Class.forName("com.testinstance.Instance");  }catch(ClassNotFoundException cnfe) { System.out.println("Class Not Found");  }  System.out.println(Instance.i);  }  } | 100 |

|  |  |  |
| --- | --- | --- |
| CoreJava- AccessSpecifiers-032 | Consider the following code: package com.testinstance;  class Instance { static int i=1000;  {  try {  Class.forName("com.testinstance.Instance"); i++;  } catch(Exception e) { System.out.println("Exception");  }  }  }  public class TestInstance {  public static void main(String args[]) { try {  Class.forName("com.testinstance.Instance").newInstance ();  }catch(Exception e) { System.out.println("Exception");  }  System.out.println(new Instance().i);  } | 1002 |

|  |  |  |
| --- | --- | --- |
| CoreJava- AccessSpecifiers-033 | Consider the following code: package com.test;  class StaticInstance { static int i = 10;  static { i+=10;  try {  Class.forName("com.test.StaticInstance").newInstance();  } catch(Exception e) { System.out.println("Class Not Found");  }  }  {  i+=10;  }  }  public class TestStaticInstance {  public static void main(String[] args) { try {  Class.forName("com.test.StaticInstance").newInstance();  } catch(Exception e) { | 40 |

|  |  |  |
| --- | --- | --- |
| CoreJava- AccessSpecifiers-034 | Consider the following code: package com.test;  class TwoStatic { static int i = 10;  static {  int i = 100;  i += StaticInstance.i; System.out.println(i);  }  static { System.out.println(i);  }  }  public class TwoStatic {  public static void main(String[] args) { System.out.println(TwoStatic.i);  }  }  Which of the following option gives the output for the above code? | 110  10  10 |

|  |  |  |
| --- | --- | --- |
| CoreJava- AccessSpecifiers-035 | Consider the following code:  public class TestClass { private int i, j, k; public TestClass() { }  private TestClass(int i, int j, int k) { this(i, j);  this.k = k;  }  protected TestClass(int i, int j) { this(i);  this.j = j;  }  TestClass(int i) { this.i = i; } private int getI() { return i; } private int getJ() { return j; } private int getK() { return k; }  public static void main(String[] args) { TestClass tc1 = new TestClass(); TestClass tc2 = new TestClass(100); TestClass tc3 = new TestClass(1, 2);  TestClass tc4 = new TestClass(10, 20, 30); System.out.println(tc2.getI()); System.out.println(tc3.getI() + ", " + tc3.getJ()); System.out.println(tc4.getI() + ", " + tc4.getJ()  + ", " + tc4.getK());  } | 3,5 |
| CoreJava- AccessSpecifiers-036 | Consider the following code:  class AllClass {  private static int i = 10; static { i += 10; }  { i += 10; }  AllClass() { i += 10; }  AllClass incrementWith10() { i += 10; return this;}  }  public class AllAccess {  public static void main(String[] args) { System.out.println(new  AllClass().incrementWith10().i);  }  }  Which of the following option gives the output for the above code? | Compile time error |

|  |  |  |
| --- | --- | --- |
| CoreJava-Annotations- 001 | Which of the following keyword is used to define user defined annotation type? | @interface |
| CoreJava-Annotations- 002 | Which of the following options give the annotation types that are predefined by the language specification itself?   1. '@Deprecated 2. '@Override 3. '@SuppressWarnings 4. '@Documented 5. '@Target | 1,2,3 |
| CoreJava-Annotations- 003 | Which of the following option gives the super type of all Annotation types? | java.lang.annotation.Annot ation |
| CoreJava-Annotations- 004 | Which of the following option gives the Java Language Element that cannot be annotated? | Try-Catch Blocks |
| CoreJava-Annotations- 005 | Which of the following option gives the value of the ElementType that controls an annotation type to be applicable only for classes, interfaces, annotation types and enums? | ElementType.TYPE |
| CoreJava-Annotations- 006 | Which of the following options give the marker annotations among built-in annotations in Java?   1. '@Target 2. '@Retention 3. '@Override 4. '@Deprecated 5. '@Inherited | 3,4,5 |
| CoreJava-Annotations- 007 | Which of the following is true about Annotations? | An annotation is a special kind of modifier |
| CoreJava-Annotations- 008 | Which of the following Java language element allows method declarations to be assigned to compile-time constants? | Annotations |

|  |  |  |
| --- | --- | --- |
| CoreJava-Annotations- 009 | Which of the following are true regarding Annotation?   1. Annotation refers to data in a Java Program 2. Annotation refers to metadata in a Java Program 3. Annotations can be embedded into class files 4. Annotations are not retrievable at run-time 5. Annotations can replace XML configurations | 2,3,5 |
| CoreJava-Annotations- 010 | Which of the following option gives the Built-in annotation, that controls the class level and runtime accessibility of an User defined annotation? | @Retention |
| CoreJava-Annotations- 011 | Which of the following option gives the Built-in annotation, that controls the use of an User defined annotation with various Java Language Elements? | @Target |
| CoreJava-Annotations- 012 | Consider the following code:  public @interface Demo { public String value();  }  Which of the following options give the code that properly uses the above defined annotation?   1. '@Demo   class MyClass { }   1. '@Demo(value = "Test") class MyClass { } 2. '@Demo("Test") class MyClass { } 3. '@Demo(value="Test"); class MyClass { } 4. '@Demo("Test"); class MyClass { } | 2,3 |

|  |  |  |
| --- | --- | --- |
| CoreJava-Annotations- 013 | Which of the following statements are true regarding the usage of @Inherited annotation type?   1. Indicates that a class or interface is extended from its super type 2. '@Inherited annotation type works only with classes 3. Indicates that a method is inherited from its super class 4. Indicates that an annotation type used for a superclass should get inherited to its sub classes also 5. Indicates that an annotation type is inherited from another annotation type | 1,4 |
| CoreJava-Annotations- 014 | Which of the following statements are true about  @Override annotation?   1. The @Override annotation is discarded by the compiler, while creating class files 2. The @Override annotation is recorded in the class file by the compiler but need not be retained by the Virutal Machine at run time 3. The @Override annotation is recorded in the class file by the compiler and retained by the Virutal Machine at run time 4. The @Override annotation is used by the compiler just to generate the error message if not properly overriden 5. The @Override annotation is used only by the IDE in order to indicate that a method is an overriden method | 1,4 |
| CoreJava-Annotations- 015 | Which of the following is true regarding the use of  @Override? | The @Override can be used only while overriding a method in the super class |

|  |  |  |
| --- | --- | --- |
| CoreJava-Annotations- 016 | Consider the following:  An annotation type need to be defined with the name 'PreInit', that is applicable only for methods and should be reflexive.  The annotation should support an integer type method called 'initValue' with the default value 0.  Which of the following code gives the correct annotation type defintion for the above specification? | @Retention(RetentionPolic y.RUNTIME)  @Target(ElementType.MET HOD)  public @interface PreInit { int initValue() default 0;  } |
| CoreJava-Annotations- 017 | Consider the following code:  @Target(ElementType.TYPE) public @interface ClipInfo {  Integer clipId(); String clipName();  String artist() default "[unknown]"; String date() default "[unknown]";  }  Which of the following code / code snippets give the proper usage of above annotation type definition?   1. '@ClipInfo(clipId=1, clipName="New Clip") interface I { } 2. '@ClipInfo(clipId=10, clipName="Old Clip") enum E { } 3. '@ClipInfo(clipId=100, clipName="Final Clip") class C { } 4. '@ClipInfo(aritst="New Artist", date="01-Jan-2010") class C { } 5. '@ClipInfo(aritst="Old Artist", date="01-Feb-2010") enum E { } | 2,3 |
| CoreJava-Collections- 001 | Consider the following Statements:  A: WeakHashMap is synchronized.  B: All Set implementation Rejects Duplicates but is ordered.  Which of the following option is true regarding the above statements? | Both the Statements A and B are false |

|  |  |  |
| --- | --- | --- |
| CoreJava-Collections- 002 | Consider the following list of items in Column A and column B  Column A   1. Vector 2. HashSet 3. TreeSet 4. ArrayList   Column B   * 1. It is not ordered or sorted   2. It compares and stores the data in a sorting order   3. Permits all elements including NULL.   4. It tries to optimize storage management by maintaining a capacity and a capacity Increment.   Which of the following option gives the valid match between the Items in Column A and Statements in Column B? | 2-A, 3-B, 4-C, 1-D |
| CoreJava-Collections- 003 | Which of the following option gives the valid collection implementation class that implements the List interface and also provides the additional methods to get, add and remove elements from the head and tail of the list without specifying an index? | LinkedList |
| CoreJava-Collections- 004 | Which of the following statements give the advantages of using a Generic Collection?   1. Thread Safety 2. Type Safety 3. JVM Safety 4. Automatic Type Casting 5. Quicker Garbage Collection | 2,4 |
| CoreJava-Collections- 005 | Which of the following statement gives the disadvantage of Generic Collection? | Not compatible with the applications developed using JDKs prior to the version 1.5 |

|  |  |  |
| --- | --- | --- |
| CoreJava-Collections- 006 | Consider the following listings: Column A:   1. Denotes a family of subtypes of type Type 2. Denotes a family of supertypes of type Type 3. Denotes the set of all types or ANY   Column B:   * 1. ? extends Type   2. ? implements Type   3. ? super Type   4. \*   5. ?   Which of the following option gives the exact match between the definitions in Column A and the syntaxes in Column B? | 1-A, 2-C, 3-E |
| CoreJava-Collections- 007 | Consider the following Statements:  Statement A: The ListIterator interface helps to traverse a list in either direction.  Statement B: hasPrevious() is a method in Iterator interface.  Which of the following option is correct regarding the above given statements? | Statement A is true and B is false |
| CoreJava-Collections- 008 | Consider the following Statements:  Statement A: The Iterator interface declares only two methods: hasMoreElements and nextElement.  Statement B: The ListIterator interface extends both the List and Iterator interfaces.  Which of the following option is correct regarding above given statements? | Both the statements A and B are false. |
| CoreJava-Collections- 009 | Consider the following partial code: java.util.Date date = new java.util.Date();  Which of the following statement is true regarding the above partial code? | Creates a Date object with current date and time as default value |

|  |  |  |
| --- | --- | --- |
| CoreJava-Collections- 010 | Consider the following code snippets:   1. java.sql.Date d = new java.sql.Date(); 2. java.util.Date d = new java.util.Date(); 3. java.util.Calendar cal = java.util.Calendar.getInstance(); java.util.Date d = cal.getDate(); 4. java.util.Calendar cal = java.util.Calendar.getInstance();   java.util.Date d = cal.getTime();   1. String strDate = "01/01/2010";   java.util.Date d = java.util.Date.parseDate(strDate);  Which of the following option giveS the valid code snippets from the above list that gives the valid way of creating a date object? | B,D |
| CoreJava-Collections- 011 | Consider the following partial code:  for (Iterator myListIterator = myList.iterator(); myListIterator.hasNext(); ) {  String myElement = (String) myListIterator.next(); System.out.println(myElement);  }  Assume that 'myList' is a java.util.List type object.  Which of the following option gives the code which is equivalent to the above partial code using Generics? | for (Iterator<String> myListIterator = myList.iterator(); myListIterator.hasNext(); ) {  String myElement = myListIterator.next();  System.out.println(myElem ent);  } |

|  |  |  |
| --- | --- | --- |
| CoreJava-Collections- 012 | Consider the following code:   1. import java.util.Set; 2. import java.util.TreeSet;   03   1. class TestSet { 2. public static void main(String[] args) { 3. Set set = new TreeSet<String>(); 4. set.add("Green World"); 5. set.add(1); 6. set.add("Green Peace"); 7. System.out.println(set); 11 }   12 }  Which of the following option gives the output for the above code? | Throws Runtime Exception |
| CoreJava-Collections- 013 | Consider the following code:  import java.util.Set; import java.util.TreeSet; import java.util.Arrays;  public class TestSet {  public static void main(String args[]) { Integer[] num = {7,5,7,3}; Set<Integer> set = new  TreeSet<Integer>(Arrays.asList(num)); set.add(4);  for(Integer value: set) { System.out.print(value + " ");  }  }  }  Which of the following option gives the valid output for the above code? | 3, 4, 5, 7 |

|  |  |  |
| --- | --- | --- |
| CoreJava-Collections- 014 | Consider the following code and predict the output: import java.util.\*;  public class TestTreeMap {  public static void main(String args[]) { TreeMap tree = new TreeMap(); tree.put("aa", "bb");  tree.put("cc", "dd");  tree.put("ee", "ff");  tree.put("gg", "hh");  Map map = tree.tailMap("ee"); System.out.println(map);  }  }  Which of the following option gives the valid output for the above code? | {ee=ff, gg=hh} |
| CoreJava-Collections- 015 | Consider the following code and predict the output: import java.util.\*;  public class SetTest{  public static void main(String[] args) { Set s = new HashSet(); s.add("Alaska"); // Line 1 s.add(new Fashion()); // Line 2  Set t = new TreeSet(); t.add("Alaska"); // Line 3 t.add(new Fashion()); // Line 4 System.out.println(s); System.out.println(t);  }  }  class Fashion { public String toString() { return "Fashion";  } }  Which of the following option is valid, if the above program is compiled and executed? | Runtime error at the line commented as Line 2 |

|  |  |  |
| --- | --- | --- |
| CoreJava-Collections- 016 | Consider the following statements about the Map type Objects:  Statement A: Changes made in the set view returned by keySet() will be reflected in the original map.  Statement B: All Map implementations keep the keys sorted.  Which of the following option is true regarding the above statements? | Statement A is true and Statement B is false |
| CoreJava-Collections- 017 | Consider the following list of code:   1. Iterator iterator = hashMap.keySet().iterator(); 2. Iterator iterator = hashMap.iterator(); 3. Iterator iterator = hashMap.keyMap().iterator(); 4. Iterator iterator = hashMap.entrySet().iterator(); 5. Iterator iterator = hashMap.entrySet.iterator();   Assume that hashMap is an instance of HashMap type collection implementation.  Which of the following option gives the correct partial code about getting an Iterator to the HashMap entries? | D |
| CoreJava-Collections- 018 | Consider the following code: import java.util.\*;  public class IterateOurSatellites {  public static void main(String[] args) { List l = new ArrayList();  l.add("GSLV"); l.add("PSLV"); l.add("SLV");   * 1. dd("Chandrayaan"); l.add("IRS"); ListIterator i = l.listIterator();      1. next(); i.next(); i.next(); i.next();i.next(); i.remove();   i.previous(); i.previous(); i.remove(); System.out.println(l);  }  }  Which of the following option gives the valid output for the above code? | [GSLV, PSLV, Chandrayaan] |

|  |  |  |
| --- | --- | --- |
| CoreJava-Collections- 019 | Consider the following code:  import java.text.ParseException; import java.text.SimpleDateFormat; import java.util.\*;  public class String2DateTest {  public static void main(String[] args) {  String dateAndTime = "10/05/2010 10:30:00 PM";  /\* CODE \*/ SimpleDateFormat sdf = new  SimpleDateFormat(dateFormat); try {  Date date = sdf.parse(dateAndTime); System.out.println(date);  }catch(ParseException pe) { System.out.println("Invalid date format");  }  }  }  Which of the following option is a valid code that can be substituted to the line commented as /\* CODE \*/ in the above program, so that it converts the String to Date properly? | String dateFormat = "dd/MM/yyy hh:mm:ss a"; |

|  |  |  |
| --- | --- | --- |
| CoreJava-Collections- 020 | Consider the following code:  import java.text.ParseException; import java.text.SimpleDateFormat; import java.util.\*;  public class Date2StringTest {  public static void main(String[] args) { Date today = new Date();  String now = "";  /\* CODE \*/ SimpleDateFormat sdf = new  SimpleDateFormat(timeFormat); now = sdf.format(today); System.out.println(now);  }  }  Which of the following option is a valid code that can be substituted to the line commented as /\* CODE \*/ in the above program, so that it prints current time properly? | String timeFormat = "hh:mm:ss a"; |
| CoreJava-Collections- 021 | Consider the following Scenario:  Jagadeesh is creating an application in which he wants to use a collection container with the following features:   1. Container should not allow duplicates 2. The container when instantiated, should be initialized with the elements of an existing Set. 3. Iteration order of the new Set should be same as that of the existing Set.   Which of the following option gives the most suitable Collection class implementation that Jagadeesh should use for building this application? | LinkedHashSet |

|  |  |  |
| --- | --- | --- |
| CoreJava-Collections- 022 | Consider the following scenario:  A java application need to load all the rows as objects from a database table and keep it in the memory, with the following features:   * An object should be fetched based on the primary key * When a new object is added to the memory, it should get inserted to the database table * When an existing object is removed, the corresponding row should be removed from the database * When any changes are made to the object the changes have to be updated to the database.   Which of the following collection implementation would be suitable for the above scenario? | HashMap |
| CoreJava-Collections- 023 | Consider the following scenario:  Two Client Systems A and B are connected to a Server. Client A keeps sending some messages asynchronously to Client B through the Server. The server keeps track of all the messages coming from Client A and dispatches it in the same sequences how it is recevied from Client A. The messages sent by Client A is delivered to Client B after a short time delay.  Which of the following collection implementation the server should use to handle the messages sent by Client A? | Queue |
| CoreJava-Collections- 024 | Consider the following scenario:  An ArrayList is popluated with the names of all the employees in a company.  Now the names start with the string "AR" should only be removed from the list.  Which of the following option gives the correct interface that helps to accomplish the above task? | Iterator |

|  |  |  |
| --- | --- | --- |
| CoreJava-Collections- 025 | Consider the following scenario:  The Employee objects are populated in an ArrayList in order to show it to the user.  The User Interface is designd in such a way to display one record at a time.  Appropriate buttons are provided to browse through the previous and next record.  Which of the following option gives the correct interface that helps to accomplish the above task? | ListIterator |
| CoreJava-Collections- 026 | Consider the following scenario:  An Online Ticket Booking application designed using Java needs to store the booked date and time.  Which of the following option gives the valid class that can accomplish the above task? | java.util.Date |
| CoreJava-Collections- 027 | Consider the following scenario:  An HR application is being developed in Java. A particular module needs to calculate the number of days between project start date and end date for an employee, in order to arrive at the billable days. The given inputs are day, month and year values for project start date and end date.  Which of the following option gives the valid class that can accomplish the above task? | java.util.Calendar |

|  |  |  |
| --- | --- | --- |
| CoreJava-Collections- 028 | Consider the following scenario:  A method called getRows() in a Java module needs to return some set of objects when the method is called, but subject to the conditions below:   * The objects can be duplicated * But the Objects should maintain the order * The number of fields in the object will vary for every method call   (For example: For the first call there could be 5 fields in the returned list of objects, and for the next call there could be 3 fields and so on.)   * Types of fields may also vary for every call.   Which of the following Generic Collection based method Declaration is most suitable for handling the data related to the above scenario effectively? | public List<Map<String, Object>> getRows(); |
| CoreJava-Collections- 029 | Consider the following scenario:  An online application maintains a list of logged-in users. Every user has some set of accounts configured by them, but subject to the following conditions:   * The Users list should be unique * The accounts list also should be unqiue * With the help of user, the corresponding accounts also should be accessible   Which of the following Generic Collection Declaration can handling the data related to the above scenario effectively? | private Map<User, Set<Account>> users = null; |

|  |  |  |
| --- | --- | --- |
| CoreJava-Collections- 030 | Consider the following scenario:  A method called getRecords() in a Java module needs to returns some set of objects when the method is called, but subject to the conditions below:   * The objects are unique * The Objects are sorted * The number of fields in the object will vary for every method call   (For example: For the first call there could be 5 fields in the returned list of objects, and for the next call there could be 3 fields and so on.)   * Types of fields may also vary for every call.   Which of the following Generic Collection based method Declaration is most suitable for handling the data related to the above scenario effectively? | public SortedSet<Map<String, Object>> getRecords(); |
| CoreJava-Collections- 031 | Consider the following code: 01 import java.util.\*;  02   1. public class Convert2Generic { 2. public static void main(String[] args) { 3. List customerList = null; 4. customerList = readCustomers(); 07 }   08   1. private static List readCustomers() { 2. // Code to read and return customers data 11   12 }  13 }   1. class Customer { 2. public String customerId; 3. public String customerName; 17 }   Which of the following options give the line number which needs to be changed in order to enable generics for the above program, so that the List should handle only the Customer class type objects?   1. The declaration at Line no. 05 2. The statement at line no. 06 3. The declaration at Line no. 09 | 1,3 |

|  |  |  |
| --- | --- | --- |
| CoreJava-Collections- 032 | Consider the following code: import java.util.\*;  public class Test {  public static void main(String[] args) {  List elements = new ArrayList(); // Line 5 elements.add(194); elements.add(34.48); elements.add((byte) 34);  elements.add((short) 234); System.out.println(elements);  }  }  Which of the following code has to be replaced for the line which is commented as // Line 5 in the above program, in order to enable Generics? | List<Number> elements = new ArrayList<Number>(); |
| CoreJava-Collections- 033 | Consider the following code: import java.util.\*;  class Person { }  class Employee extends Person { } class Consultant extends Person { }  public class TestGenericWildCard { public static void main(String[] args) {  List<Person> persons = new ArrayList<Person>(); persons.add(new Person());  persons.add(new Person()); process(persons);  List<Employee> employees = new ArrayList<Employee>();  employees.add(new Employee()); employees.add(new Employee()); process(employees);  List<Consultant> consultants = new ArrayList<Consultant>();  consultants.add(new Consultant()); consultants.add(new Consultant()); process(consultants);  } | 1,2 |

|  |  |  |
| --- | --- | --- |
| CoreJava-Collections- 034 | Consider the following Code:  import java.util.List; import java.util.ArrayList;  import java.util.Collections;  public class SearchCollection {  public static void main(String[] args) { List<String> techs = new ArrayList<String>(); techs.add("java");  techs.add("flex");  techs.add("flash"); Collections.sort(techs);  System.out.print(Collections.binarySearch(techs, "flex") + " ");  System.out.print(Collections.binarySearch(techs, "c#"));  }  }  Which of the following will be the valid output for the above code? | Prints: 1 -1 |
| CoreJava-Collections- 035 | Consider the following code: import java.util.\*;  public class ListListList {  public static void main(String[] args) { List list = new ArrayList(); list.add("1");  list.add("2");  list.add(1, "3"); list.remove(2);  List linkedlist = new LinkedList(list); list.addAll(linkedlist);  linkedlist = list.subList(0,3); linkedlist.clear(); System.out.println(list);  }  }  Which of the following option gives the valid output for the above code? | [3] |

|  |  |  |
| --- | --- | --- |
| CoreJava-Collections- 036 | Consider the following code:  import java.util.ArrayList; import java.util.List;  public class FriendsList {  private List<Friend> friendsList = null;  public FriendsList() {  /\* CODE \*/  }  public void addFriend(Friend friend) {  // Add friend logic  }  public void removeFriend(String friendName) {  // Remove friend logic  }  }  class Friend {  private String friendName;  public Friend(String friendName) { this.friendName = friendName;  }  }  The above class maintains the friends list. It has two business methods namely addFriend and removeFriend. | 2,3 |
| CoreJava-Collections- 037 | Consider the following code:   1. class TestIndexedIterator { 2. public static void main(String args[]) { 3. List<String> myList = new ArrayList<String>(); 4. myList.add("Windows"); 5. myList.add("Linux"); 6. myList.add("Mac OS"); 7. int index=0; 8. for (Iterator<String> myListiterator = myList.iterator(); myListiterator.hasNext();) { 9. System.out.println(myList.get(index++)); 10 }   11 }  12 }  Which of the following option gives the correct output for the above code? | Prints the following with a RuntimeError at the end  Windows Linux Mac OS |

|  |  |  |
| --- | --- | --- |
| CoreJava-Collections- 038 | Consider the following code: import java.util.\*;  public class Test {  public static void main(String[] args) { List<String> myList = new ArrayList<String>(); myList.add("Chennai");  myList.add("Delhi"); myList.add("Pune"); myList.add("Mumbai"); myList.add("Kolkata");  Iterator<String> iStr = myList.iterator(); while(iStr.hasNext()) {  String element = iStr.next(); if(element.equals("Pune")) { myList.remove(element);  }  }  for(String element : myList) { System.out.print(element + " ");  }  }  }  Which of the following option gives the valid output for | Runtime Exception |

|  |  |  |
| --- | --- | --- |
| CoreJava-Collections- 039 | Consider the following incomplete code:  import java.util.Calendar;  import java.util.GregorianCalendar; import java.util.Date;  public class PriorAhead {  public static void main(String[] args) {  Calendar cal = GregorianCalendar.getInstance(); cal.setTime(new Date());  /\* LINE 1 \*/ System.out.println(cal.getTime());  /\* LINE 2 \*/ System.out.println(cal.getTime());  }  }  Which of the following option when substituted to the comments /\* LINE 1 \*/ and /\* LINE 2 \*/ in the above code will print the date which is 120 days prior the current and 120 days ahead of current date? | LINE 1:  cal.add(Calendar.DAY\_OF\_ MONTH, -120);  LINE 2:  cal.add(Calendar.DAY\_OF\_ MONTH, 240); |

|  |  |  |
| --- | --- | --- |
| CoreJava-Collections- 040 | Consider the following partial code:  import java.util.Calendar;  import java.util.GregorianCalendar; import java.util.Date;  public class FiveDaysAhead {  public static void main(String[] args) {  Calendar cal = GregorianCalendar.getInstance(); cal.setTime(new Date());  /\* LINE 1 \*/ System.out.println(cal.getTime());  }  }  Which of the following options when substituted to the comment /\* LINE 1 \*/ in the above code will print the date which is 5 days ahead of current date?   1. cal.add(Calendar.DAY, 5); 2. cal.roll(Calendar.DAY\_OF\_MONTH, 5); 3. cal.set(Calendar.DAY, 5); 4. cal.roll(Calendar.DAY, 5); 5. cal.add(Calendar.DAY\_OF\_MONTH, 5); | 2,5 |
| CoreJava-Controlflow- 001 | Consider the following code snippet:  int i=0; while(i-- >0) {  System.out.println(“the value of i is “+i);  }  System.out.println(“Finished”);  Which of the following will be the output for the above code snippet? | Prints the message "Finished" |
| CoreJava-Controlflow- 002 | Which of the following statements are true about Integer wrapper class?   1. The Integer can accept a String in its constructor 2. The Integer has a doubleValue() method 3. The immediate super class of Integer class is java.lang.Object 4. Integer is a mutable class 5. The Integer class provides a method to convert an Integer to Binary | 1,2,5 |

|  |  |  |
| --- | --- | --- |
| CoreJava-Controlflow- 003 | Which of the following option gives the methods which are not available in Float wrapper type? | toOctalString() |
| CoreJava-Controlflow- 004 | Which of the following returns a primitive data type?   1. Integer.parseInt() 2. Integer.getInteger() 3. Integer.valueOf() 4. Integer.intValue() 5. Integer.decode() | 1,4 |
| CoreJava-Controlflow- 005 | Which of the following operators on using only with Numeric Wrapper type objects requires unboxing?  1) =  2) ==  3) ++  4) --  5) + | 3,4,5 |
| CoreJava-Controlflow- 006 | Which of the following listed loop type in Java does not depend on Boolean expression to terminate its execution? | for each loop |
| CoreJava-Controlflow- 007 | Consider the following code snippet:  public static void main(String args[]) { int N;  N=1;  While(N<=32) { N= 2\*N;  System.out.print(N + " ");  }  }  Which of the following option gives the output for the above code? | 2 4 8 16 32 64 |
| CoreJava-Controlflow- 008 | Which of the following keyword communicates information between to methods in a Java Program? | return |

|  |  |  |
| --- | --- | --- |
| CoreJava-Controlflow- 009 | Which of the following statements are true regarding switch-case structure?   1. Switch case can compare integer types 2. Break statement is important for every case in a switch-case structure, in order to get a error-free compilation 3. The default case can be kept anywhere in a switch- case structure 4. All cases including default is optional inside a switch- case structure 5. Switch case cannot compare character type | 1,3,4 |
| CoreJava-Controlflow- 010 | Which of the following statements are true regarding if- else structure and switch-case structure?   1. All logics that are implemented with if-else statement is possible to implement in switch-case structure also 2. Only equality check can be done with switch-case structure 3. If-else structure can be nested inside a switch-case structure and vice-versa 4. The case statement in the switch-case structure is an executable statement 5. Execution control can be manually transfered to any of the case inside a switch-case structure, using labled break statement | 2,3 |
| CoreJava-Controlflow- 011 | Which of the following option is not a valid wrapper type object creation? | new Character("a"); |
| CoreJava-Controlflow- 012 | Consider the following code:  class TestWrapper {  public static void main(String[] args) { boolean oldTruth = true;  Boolean newTruthOne = new Boolean("true"); Boolean newTruthTwo = new Boolean("true"); System.out.println((oldTruth == newTruthOne) ?  "new truth one" : "old truth"); System.out.println((newTruthOne == newTruthTwo) ?  "new truth two" : "new truth one");  }  }  Which of the following will be the output for the above code? | new truth one new truth one |

|  |  |  |
| --- | --- | --- |
| CoreJava-Controlflow- 013 | Consider the following code:  class TestReturn {  public static void main(String args[]) { int i = 10;  System.out.println("Result:" + getSquare(i));  }  public static int getSquare(int i) { return i \* i;  System.out.println("End of getSquare");  }  }  Which of the following option gives the output for the above code? | Compilation Error |
| CoreJava-Controlflow- 014 | Consider the following variable declarations:  int x=100, y=200; Integer i=100, j=200;  Which of the following options are true regarding the above code snippet?   1. The expressions (x == y) and (i == j) are functionally same 2. The expressions (x < y) and (i < y) are functionally same 3. The expressions (x != j) and (i != y) evaluates to the same result 4. The expression (j - i) evaluates to another Integer wrapper type object with the value 100 5. x.equals(j) is valid expression | 1,2,3 |
| CoreJava-Controlflow- 015 | Consider the following code snippet:  Integer i=1411, j=i; System.out.println(i++); System.out.println(++j);  Which of the following option gives the correct number of Auto-boxing and Auto-Unboxing occurred in the above code? | Auto-boxing: 3, Auto-  Unboxing: 2 |
| CoreJava-Controlflow- 016 | Which of the following statement is false about for-each loop in Java? | for-each loop can work only with generic collections |

|  |  |  |
| --- | --- | --- |
| CoreJava-Controlflow- 017 | Which of the following statement gives the valid condition in order to make a for-each loop to work with non-generic collections? | The variable used for iterating should of java.lang.Object type |
| CoreJava-Controlflow- 018 | Which of the following statements are true about labeled break?   1. labeled break statement in java is equivalent to goto statement 2. labels declared for single line statements cannot be used with labeled break 3. labeled break can be used to form a loop in combination with if statement 4. labeled break statement is a non executable statement 5. labeled break cannot be used with switch ... case structure | 2,5 |
| CoreJava-Controlflow- 019 | Consider the following code snippet:  Object n = 1411; System.out.println(n);  System.out.println(n.getClass().getName());  Which of the following will be the output for the above code snippet? | 1411  java.lang.Integer |

|  |  |  |
| --- | --- | --- |
| CoreJava-Controlflow- 020 | Consider the following code:   1. class TestWhileFor { 2. public static void main(String args[]) { 3 int i=0, j=1; 3. while (i==0) { 4. do   6 i++;  7 while(i<j);  8 j = i+1;  9 for(;i>0;i--)  10 System.out.println(i + " " + j); 11 }  12 }  13 }  Which of the following statements are true regarding the above code?   1. The for-loop in the line numbers 9 and 10 runs for j-i number of times 2. The for-loop in the line numbers 9 and 10 runs for j+i number of times 3. The for-loop in the line numbers 9 and 10 runs for (j- i)+1 number of times 4. The output is 3 2 | 3,4 |

|  |  |  |
| --- | --- | --- |
| CoreJava-Controlflow- 021 | Consider the following code:   1. class TestForEach { 2. public static void main(String args[]) { 3. Object list[] = {"Save Trees", new Date()}; 4. for (String o : list) 5. System.out.println(o); 6 }   7 }  Which of the following options give the valid exclusive changes that can be made to the above program to make it compile and execute properly?   1. Changing the array element, new Date() into new Date().toString(), in line number 3 2. Declaring the array as String array instead of Object array in the line number 3 3. Declaring the object o as Object instead of String in the line number 4 4. list has to be declared as collection object instead of array in line number 3 5. The object o has to be downcasted to java.lang.Object type before printing it, in line number 5 | 2,3 |

|  |  |  |
| --- | --- | --- |
| CoreJava-Controlflow- 022 | Consider the following code:  class SaveOurTigers {  public static void main(String args[]) { int count = 1411;  switch(count) {  case 1411: again: { System.out.println("Save Our Tigers"); break again;  }  default: { System.out.println("Share this"); break;  }  }  }  }  Which of the following option gives the output for the above code? | Prints:  Save Our Tigers Share this |
| CoreJava-Controlflow- 023 | Consider the following code:  class TestWrapper2 {  public static void main(String args[]) { Integer i = 1411;  Integer j = i; i++;  System.out.println((i == j) && (i.equals(j))); i--;  System.out.println((i == j) && (i.equals(j)));  }  }  Which of the following option gives the output for the above code? | false false |

|  |  |  |
| --- | --- | --- |
| CoreJava-Controlflow- 024 | Consider the following code:  public class TestVarArgs {  public static void printValues(int... x) { System.out.print("Var-args");  }  public static void printValues(Long x, Long y) { System.out.print("Long Wrapper");  }  public static void printValues(Number x, Number y) { System.out.print("Wrapper Base");  }  public static void main(String[] args) { printValues(10, 20);  }  }  Which of the following option gives the output for the above code? | Wrapper Base |
| CoreJava-Controlflow- 025 | Consider the following code:  class TestSwitchCase {  public static void main(String args[]) { int a=30;  switch(a) {  case 10: case 20:  if (a==10) System.out.print("R"); else break;  case 30: case 40:  if (a==30) System.out.print("G"); else break;  case 50: case 60:  if (a==50) System.out.print("B"); else break;  }  }  }  Which of the following will be the output for the above code? | G |

|  |  |  |
| --- | --- | --- |
| CoreJava-Controlflow- 026 | Consider the following code:  class GradeCalculator {  public static void main(String args[]) { int total = 35;  if(total <= 100) { if(total < 90) {  if(total < 75) { if(total < 60) {  if(total < 40) { System.out.println("F");  }  System.out.println("B");  }  System.out.println("A");  }  System.out.println("D");  }  System.out.println("O");  }  }  }  Which of the following statement is true regarding the above program code? | Inserting a label before the beginning of outermost if- statement and a labeled break statement after every System.out.println() statement except the outermost if-statement will make the program to correctly calculate the grade |
| CoreJava-Controlflow- 027 | Consider the following code:  class CarefulFor {  public static void main(String args[]) { int i, j, k;  for(i=0; i<3; i++); for(j=i; j<3; j++);  for(k=j; k<3; k++); System.out.print(i);  }  }  Which of the following option gives the output for the above code? | 3 |

|  |  |  |
| --- | --- | --- |
| CoreJava-Controlflow- 028 | Consider the following code:   1. class TestWhile2 { 2. public static void main(String args[]) { 3 int i = 1, j = 2, k = 3;   4   1. while (i < j) 2. while (j < k) 3. while (k < 4) 4. System.out.println(i++ + " " + j++ + " " + k++); 9 }   10 }  Which of the following statements are true regarding the above code?   1. Prints: 1 2 3 2. Prints: 2 3 4 3. Forms an indefinite loop at line number 5 4. Forms an indefinite loop at line number 6 5. Forms an indefinite loop at line number 7 | 1,4 |
| CoreJava-Controlflow- 029 | Consider the following code:  class TestWhile2 {  public static void main(String args[]) { String roar = "Save Tigers";  int index = 0;  while(index \* 2 + 1 < roar.length()) System.out.print(roar.charAt(index++ \* 2 + 1));  }  }  Which of the following gives the output for the above code? | Prints: aeTgr |

|  |  |  |
| --- | --- | --- |
| CoreJava-Exceptions- 001 | Which of the following statement is true regarding throws declaration in Exception handling? | None of the listed options |
| CoreJava-Exceptions- 002 | Which of the following statement is true regarding the throws declarations for overriden methods? | The overriding method cannot declare additional exceptions which is not declared in its super class version. |
| CoreJava-Exceptions- 003 | Which of the following listed type cannot be caught and handled using catch block? | None of the listed options |
| CoreJava-Exceptions- 004 | Which of the following options give the types that Throwable Constructor can accept?   1. String 2. StringBuffer 3. Throwable 4. Integer 5. Boolean | 1,3 |

|  |  |  |
| --- | --- | --- |
| CoreJava-Exceptions- 005 | Which of the following statements are true regarding try- catch-finally blocks?   1. Catch block is optional when a RuntimeException is thrown from the try block 2. Catch block is optional, if finally block is available, provided a Checked Exception is thrown from the try block 3. Finally block is optional, irrespective of catch block 4. Both catch block and finally block are optional, either Checked exception or unchecked exception is thrown from the try block 5. Both catch block and finally block are required, irrespective of the type of exceptions thrown from the try block | 1,3 |
| CoreJava-Exceptions- 006 | Which of the following statements are true regarding try- catch-finally?   1. An exception which is not handled by a catch block can be handled by writing another try catch block inside finally block 2. An exception which is not handled by a catch block will be handled by subsequent catch blocks 3. A catch block can have another try block nested inside 4. Finally block cannot have a try block with multiple catch blocks 5. Both catch block and finally block can throw exceptions | 2,3,5 |
| CoreJava-Exceptions- 007 | Which of the following are true regarding implementing user defined exception mechanisms?   1. It is not valid to derive a class from java.lang.Throwable 2. It is not valid to derive a class from java.lang.Error 3. It is valid to derive a class from java.lang.Exception 4. It is valid to derive a class from java.lang.RuntimeException 5. It is not valid to derive a class from java.io.IOException | 3,4 |

|  |  |  |
| --- | --- | --- |
| CoreJava-Exceptions- 008 | Which of the following are true regarding RuntimeException?   1. RuntimeException does not require a throws declaration 2. If RuntimeException is declared using throws clause, then the calling method should handle it using try-catch block 3. Any class that derives the RuntimeException will always be an unchecked exception 4. RuntimeException can be handled using a catch that handles Error 5. RuntimeException can be handled using a catch that handles Exception | 1,3,5 |
| CoreJava-Exceptions- 009 | Which of the following statements about the printStackTrace() method are true?   1. Starts the error report from the calling method 2. Starts the error report from the method whether the exception occurred 3. Reports the line number in each method from where the exception is propagated to next level 4. Stack Trace can be shown only to the console 5. The printStackTrace() method is defined in java.lang.Exception class | 2,3 |

|  |  |  |
| --- | --- | --- |
| CoreJava-Exceptions- 010 | Consider the following code:  class LastException extends Exception { }  class PreviousException extends LastException { } class TopException extends PreviousException { }  public class ExceptionalWorld {  public static void main(String args[]) { try {  exceptionThrower();  }  catch(LastException le) {System.out.println("Last Exception");}  catch(PreviousException pe)  {System.out.println("Previous Exception");}  }  static void exceptionThrower() throws TopException { throw new TopException();  }  }  Which of the following option gives the output for the above code? | Compile Time Error |
| CoreJava-Exceptions- 011 | Consider the following code:  class MyException extends Throwable { } public class TestThrowable {  public static void main(String args[]) { try {  test();  } catch(Throwable ie) { System.out.println("Exception");  }  }  static void test() throws Throwable { throw new MyException();  }  }  Which of the following option gives the output for the above code? | Prints Exception |

|  |  |  |
| --- | --- | --- |
| CoreJava-Exceptions- 012 | Consider the following code: class MyError extends Error { }  public class TestError {  public static void main(String args[]) { try {  test();  } catch(Error ie) { System.out.println("Error caught");  }  }  static void test() throws Error { throw new MyError();  }  }  Which of the following option gives the output for the above code? | Prints  Error caught |
| CoreJava-Exceptions- 013 | Consider the following code:  public class TestPropagator {  public static void main(String args[]) { try {  propagator1();  } catch(IndexOutOfBoundsException i) { System.out.println(i.getMessage());  }  }  static void propagator1() { propagator2(); } static void propagator2() { propagator3(); } static void propagator3() {  throw new StringIndexOutOfBoundsException("StringIndexOutOfBo undsException");  }  }  Which of the following option gives the output for the above code? | StringIndexOutOfBoundsEx ception |

|  |  |  |
| --- | --- | --- |
| CoreJava-Exceptions- 014 | Consider the following code:   1. public class FinallyCatch { 2. public static void main(String args[]) { 3. try { 4. throw new java.io.IOException(); 5 }   6 }  7 }  Which of the following is true regarding the above code? | Demands a finally block at line number 5 |
| CoreJava-Exceptions- 015 | Consider the following code:  public class FinallyFinallyFinally { public static void main(String args[]) {  try {  try {  throw new java.io.IOException();  }  finally { System.out.println("Finally Inner");}  } catch(Exception e){ System.out.println("Exception Outer"); }  finally { System.out.println("Finally Outer");}  }  }  Which of the following option gives the output for the above code? | Finally Inner Exception outer Finally Outer |
| CoreJava-Exceptions- 016 | Consider the following code:  public class ThrowableError {  public static void main(String args[]) { try {  throw new Throwable();  }catch(Error e) { System.out.println("Problem found");  }  }  }  Which of the following option gives the output for the above code? | Compile time error |

|  |  |  |
| --- | --- | --- |
| CoreJava-Exceptions- 017 | Consider the following code:  public class ErrorWorld {  public static void main(String args[]) { try {  errorThrower();  }catch(Error e) { System.out.println("Error caught");  }  }  public static void errorThrower() throws Error { throw new Error();  }  }  Which of the following will be the output for the above code? | Prints "Error caught" |
| CoreJava-Exceptions- 018 | Consider the following code:  public class ProblemsWorld {  public static void main(String args[]) { try {  xpect();  } catch(IOException e) { System.out.println("xpected caught");  }  }  public static void xpect() throws IOException { throw new FileNotFoundException();  }  }  Which of the following statement is true regarding the above code? | Compiles and Runs successfully and prints "xpected caught" |

|  |  |  |
| --- | --- | --- |
| CoreJava-Exceptions- 019 | Consider the following code:  class PlanetException extends Exception { }  class EarthException extends PlanetException { }  class Planet {  void revolve() throws PlanetException { System.out.println("Planet revolves");  }  }  class Earth extends Planet { void revolve() /\* CODE 1 \*/ {  System.out.println("Earth revolves");  }  }  public class WorldOfExceptions { public static void main(String args[]) {  Planet planet = new Earth(); try {  planet.revolve();  }catch(/\* CODE 2 \*/ e) { System.out.println("Problem found");  }  }  } | 2,3 |

|  |  |  |
| --- | --- | --- |
| CoreJava-Exceptions- 020 | Consider the following code: import java.io.IOException; import java.sql.SQLException;  interface Interfaze1 {  void behaviour() throws IOException;  }  interface Interfaze2 {  void behaviour() throws SQLException;  }  class Implementor implements Interfaze1, Interfaze2 {  /\* CODE \*/  }  public class TestImplementor {  public static void main(String args[]) { try {  Implementor impl = new Implementor(); impl.behaviour();  }catch(Exception e) { System.out.println("Exception caught");  }  }  } | public void behaviour() {  System.out.println("Behavi our Implemented");  } |

|  |  |  |
| --- | --- | --- |
| CoreJava-Exceptions- 021 | Consider the following code:  import java.io.FileNotFoundException; import java.io.IOException;  import java.sql.SQLException;  class SuperClass {  void method() throws IOException { System.out.println("Super Class");  }  }  class SubClass extends SuperClass {  /\* CODE \*/  }  public class TestSuper {  public static void main(String args[]) { try {  SuperClass s = new SubClass(); s.method();  }catch(Exception e) { System.out.println("Exception caught");  }  }  } | 1,2,3 |

|  |  |  |
| --- | --- | --- |
| CoreJava-Exceptions- 022 | Consider the following code:  public class TestFinallyTry {  public static void main(String args[]) { try {  throw new NullPointerException();  } catch(IndexOutOfBoundsException e) { System.out.println("Exception caught");  } finally { try {  System.out.println("Test");  } catch(NullPointerException e) {  System.out.println("NullPointerException caught");  }  }  }  }  Which of the following option gives the output for the above code? | Prints:  Test  then throws NullPointerException on the console |
| CoreJava-Exceptions- 023 | Consider the following code:  public class CheckedUnchecked { public static void main(String args[]) {  try {  System.out.println("try");  } catch(/\* CODE \*/ e) { System.out.println("Exception");  }  }  }  Which of the following code snippets when replaced for the comment /\* CODE \*/ in the above program will compile and execute properly?   1. java.lang.Error 2. java.io.IOException 3. java.lang.Throwable 4. java.lang.Exception 5. java.sql.SQLException | 1,3,4 |

|  |  |  |
| --- | --- | --- |
| CoreJava-Exceptions- 024 | Consider the following code:  class UniverseException extends Exception { }  class PlanetXException extends UniverseException { } class PlanetYException extends UniverseException { }  public class ExceptionWorld {  public static void main(String args[]) { try {  planetXExceptionThrower(); planetYExceptionThrower();  }  catch(PlanetXException px) { System.out.println("Problem in PlanetX"); }  catch(PlanetYException py) { System.out.println("Problem in PlanetY"); }  catch(UniverseException ue) { System.out.println("Problem in Universe"); }  }  static void planetXExceptionThrower() throws UniverseException {  throw new PlanetXException();  }  static void planetYExceptionThrower() throws UniverseException {  throw new PlanetYException(); | Prints:  Problem in PlanetX |

|  |  |  |
| --- | --- | --- |
| CoreJava-Exceptions- 025 | Consider the following code:  class AllException extends Exception { }  class SpecificException1 extends AllException { } class SpecificException2 extends AllException { }  public class SpecificExceptionWorld { public static void main(String args[]) {  try {  specifcXception1Thrower(); specifcXception2Thrower();  }  catch(SpecificException1 sp1) { System.out.println("Specific Problem 1"); }  catch(SpecificException2 sp2) { System.out.println("Specific Problem 2"); }  }  static void specifcXception1Thrower() throws AllException {  throw new SpecificException1();  }  static void specifcXception2Thrower() throws AllException {  throw new SpecificException2();  } | Compilation Error |

|  |  |  |
| --- | --- | --- |
| CoreJava-Exceptions- 026 | Consider the following code:   1. abstract class SuperException extends Throwable { } 2. class SpecialException extends SuperException { } 3 3. public class TestAbstractThrowable { 4. public static void main(String args[]) { 5. try { 6. testAbstract(); 7. } catch(SuperException ae) { 8. System.out.println("SuperException caught"); 10 }   11 }  12   1. static void testAbstract() { 2. throw new SpecialException(); 15 }   16 }  Which of the following statements are true regarding the above code?   1. Shows Unreachable catch block for SuperException, at line number 8 2. Shows Unhandled exception type SpecialException, at line number 13 3. No error in the code | 1,5 |

|  |  |  |
| --- | --- | --- |
| CoreJava-Exceptions- 027 | Consider the following code:   1. abstract class TopException extends RuntimeException { } 2. public class TestAnonymousException { 3. public static void main(String args[]) { 4. try { 5. testAbstract(); 6. } catch(Exception ae) { 7. System.out.println("TopException caught"); 8 }   9 }  10   1. static void testAbstract() throws Exception { 2. throw new TopException() { }; 13 }   14 }  Which of the following option gives the output for the above code? | No error in the code Prints:  TopException caught |

|  |  |  |
| --- | --- | --- |
| CoreJava-Exceptions- 028 | Consider the following code:  public class TestRecException {  public static void main(String args[]) { testRecException(3);  }  static void testRecException(int n) throws RuntimeException {  try {  if(n == 3) throw new  NullPointerException("NullPointerException"); else if(n == 2)  throw new IllegalArgumentException("IllegalArgumentException");  else if(n == 1) throw new  IndexOutOfBoundsException("IndexOutOfBoundsExcepti on");  testRecException(n - 1);  } catch(RuntimeException re) { System.out.println(re.getMessage());  } finally { System.out.println("finally");  }  } | NullPointerException finally |

|  |  |  |
| --- | --- | --- |
| CoreJava-Exceptions- 029 | Consider the following code:  public class TestRecursiveRuntime { public static void main(String args[]) {  testRecRuntime(3);  }  static void testRecRuntime(int n) throws RuntimeException {  try {  if(n > 0)  testRecRuntime(n - 1);  if(n == 0) throw new  RuntimeException("RuntimeException"); else if(n == 1)  throw new NullPointerException("NullPointerException");  else if(n == 2) throw new  IndexOutOfBoundsException("IndexOutOfBoundsExcepti on");  else if(n == 3) throw new  IllegalArgumentException("IllegalArgumentException");  } catch(IllegalArgumentException ie) { System.out.println("IllegalArgumentException"); | RuntimeException NullPointerException IndexOutOfBoundsExceptio n  IllegalArgumentException |

|  |  |  |
| --- | --- | --- |
| CoreJava-Exceptions- 030 | Consider the following code:  public class TestRecRuntimeException { public static void main(String args[]) {  testRecRuntime(3);  }  static void testRecException(int n) throws RuntimeException {  try {  if(n > 0)  testRecException(n - 1);  throw new RuntimeException("RuntimeException");  } catch(RuntimeException re) { System.out.println("RuntimeException");  }  }  }  Which of the following option gives the output for the above code? | RuntimeException RuntimeException RuntimeException RuntimeException |

|  |  |  |
| --- | --- | --- |
| CoreJava-Exceptions- 031 | Consider the following code:  public class HierarchyCommunicator { public static void main(String args[]) {  try {  communicator1();  } catch(Throwable e) {  System.out.println("ArrayIndexOutOfBounds caught");  }  }  static void communicator1() throws /\* CODE 1 \*/ { communicator2(); }  static void communicator2() throws /\* CODE 2 \*/ { communicator3(); }  static void communicator3() throws /\* CODE 3 \*/ { throw new ArrayIndexOutOfBoundsException();  }  }  Which of the following code snippets when replaced to those comments /\* CODE 1 \*/ /\* CODE 2 \*/ and /\* CODE 3 \*/ in the above code will make the program to generate the output "ArrayIndexOutOfBounds caught"?  1) CODE 1 - Exception  CODE 2 - RuntimeException | 1,2 |

|  |  |  |
| --- | --- | --- |
| CoreJava-Exceptions- 032 | Consider the following code:  import java.io.IOException;  import java.io.FileNotFoundException;  public class TestThrowInFinally {  public static void main(String args[]) { try {  method1();  } catch(Exception e) { System.out.println(e.getMessage());  }  }  static void method1() throws IOException { try {  method2();  } finally {  System.out.println("finally method1"); throw new IOException("IOException");  }  }  static void method2() throws FileNotFoundException { throw new  FileNotFoundException("FileNotFoundException");  }  } | finally method1 IOException |

|  |  |  |
| --- | --- | --- |
| CoreJava-Exceptions- 033 | Consider the following code:   1. public class TestReturnInFinally { 2. public static void main(String args[]) { 3. System.out.println(testMethod()); 04 } 4. static void testMethod() { 5. try { 6. try { 7. System.out.println("Returning from inner try"); 8. return 100; 9. } finally { 10. System.out.println("Returning from inner finally"); 11. return 200;   13 }   1. } finally { 2. System.out.println("Returning from outer finally"); 3. return 300;   17 }  18 }  19 }  Which of the following option gives the output for the above code? | Returning from inner try Returning from inner finally Returning from outer finally 300 |
| CoreJava- GarbageCollection-001 | Consider the following code snippet:  String message = "What Reference Type is this?"; Which of the following option is the name of the  Reference type that is shown in the above code snippet? | Strong Reference |
| CoreJava- GarbageCollection-002 | Consider the following code snippet: Object testObject = new Object();  Which of the following option gives the code that can be used to destroy an object from the memory? | Only the Garbage Collection System can destroy an object. |

|  |  |  |
| --- | --- | --- |
| CoreJava- GarbageCollection-003 | Which of the following statements are true about Weak Reference?   1. Weak References are cleared aggressively 2. Weak References will stay in memory for a while 3. Checked by the garbage collector before throwing OutOfMemoryError 4. WeakReference is the base class for the other two references 5. Objects in Weak reference can be processed even after they become unreachable | 1,5 |
| CoreJava- GarbageCollection-004 | Which of the following statements are true about Soft Reference?   1. Soft References are created aggressively 2. Soft References are kept for a while in the memory 3. SoftReference extends WeakReference 4. Soft References can be used for implementing memory caches 5. Soft References internally keeps a second copy of every reference it maintains | 2,4 |
| CoreJava- GarbageCollection-005 | The garbage collector makes sure that all  objects held by soft references are garbage collected before the VM throws an OutOfMemoryError. | Soft Reference |
| CoreJava- GarbageCollection-006 | Which of the following statements are true about finalize method?   1. finalize will always run before an object is garbage collected 2. finalize may run before or after an object is garbage collected 3. finalize will run when an object becomes unreachable 4. finalize allows a programmer to free memory allocated to an object 5. finalize method will be called only once by the garbage collector | 1,5 |

|  |  |  |
| --- | --- | --- |
| CoreJava- GarbageCollection-007 | Which of the following happens if an uncaught exception is thrown from during the execution of the finalize() method of an object? | The exception will be ignored and the garbage collection (finalization) of that object terminates |
| CoreJava- GarbageCollection-008 | Which of the following statement is true about Phantom References? | Enqueued only when the object is physically removed |
| CoreJava- GarbageCollection-009 | Which of the following class is used by WeakReference class in order to collect the dead references? | java.lang.ref.ReferenceQue ue |
| CoreJava- GarbageCollection-010 | Consider the following code: import java.util.\*;  class Customer {  private String customerName;  public Customer(String customerName) { this.customerName = customerName;  }  public void setCustomerName(String customerName) { this.customerName = customerName;  }  }  public class GCTest2 {  public static void main(String[] args) { List<Customer> customers = new  ArrayList<Customer>();  Customer customer1 = new Customer("A"); Customer customer2 = new Customer("B"); Customer customer3 = new Customer("C"); customers.add(customer1); customers.add(customer2); customers.add(customer3);  customer1 = customer2 = customer3 = null; customers.get(2).setCustomerName("Z");  } | 1 object |

|  |  |  |
| --- | --- | --- |
| CoreJava- GarbageCollection-011 | Consider the following code: import java.util.\*;  public class GCTest3 {  public static void main(String[] args) { Integer number1 = 100;  Integer number2 = number1++;  Integer number3 = number1 + number2; System.out.println("number1:" + number1); System.out.println("number2:" + number2); System.out.println("number3:" + number3);  }  }  Which of the following option gives the valid number of objects that becomes eligible for garbage collection in the above program? | 1 object |
| CoreJava- GarbageCollection-012 | Consider the following code:  public class GCTest4 {  public static void main(String[] args) { String firstString = "Always Stable"; String secondString =  firstString.substring(firstString.indexOf(firstString), firstString.length());  /\* CODE \*/  }  }  Which of the following code snippet when substituted at the line commented as /\* CODE \*/ will make the String object "Always Stable", eligible for garbage collection? | None of the options is required, the substring() method returns null, so it makes the string eligible for garbage collection. |

|  |  |  |
| --- | --- | --- |
| CoreJava- GarbageCollection-013 | Consider the following code:  public class Test {  public static void main(String[] args) { String string = "String";  StringBuffer stringBuffer = new StringBuffer(string); StringBuilder stringBuilder = new StringBuilder(string);  /\* CODE \*/  }  }  Which of the following code snippet when substituted at the line commented as /\* CODE \*/ in the above program will make the String object "String", eligible for garbage collection? | string = null; |
| CoreJava- GarbageCollection-014 | Consider the following code:  public class Island { Island i;  public void display() { System.out.println("island"); this.i.display();  }  public static void main(String [] args) { Island i2 = new Island();  Island i3 = new Island(); Island i4 = new Island(); i2.i = i3;  i3.i = i4; i4.i = i2;  i3 = i4 = null; i2.display(); // CODE  }  }  Which of the following statements regarding the above code are true?  1) On executing the above program it prints: island  island Island | 2,3 |

|  |  |  |
| --- | --- | --- |
| CoreJava- GarbageCollection-015 | Consider the following code:  public class EnsureGC{  Integer number1 = new Integer(100); Integer number2 = new Integer(200); Integer number3 = new Integer(300);  public static void main(String argv[]){ EnsureGC egc = new EnsureGC(); egc.ensure();  }  public void ensure(){ System.out.println(number1); System.out.println(number2); System.out.println(number3);  }  }  Which of the following option ensures that the Integer objects are garbage collected at a particular point in this code? | None of the listed options |
| CoreJava- GarbageCollection-016 | Consider the following code:  Line No. 01 public class TellMeWhere { Line No. 02 StringBuilder sbuilder;  Line No. 03 public static void main(String argv[]){ Line No. 04 TellMeWhere tmw = new TellMeWhere();  Line No. 05 tmw.tell(); Line No. 06 }  Line No. 07 public void tell() {  Line No. 08 sbuilder = new StringBuilder("First time");  Line No. 09 StringBuilder sbuilder2 = sbuilder; Line No. 10 StringBuilder sbuilder3 = new StringBuilder("Second Time");  Line No. 11 sbuilder=sbuilder3; Line No. 12 sbuilder3=null; Line No. 13 sbuilder2=null; Line No. 14 }  Line No. 15 }  Which of the following option gives the correct line number at which the object created on Line No. 08 will be eligible for garbage collection? | Line No. 13 |

|  |  |  |
| --- | --- | --- |
| CoreJava- GarbageCollection-017 | Consider the following code: import java.util.\*;  public class GCTest5 {  public static void main(String[] args) { HashSet<String> h = new HashSet<String>(); h.add(new String("Readable"));  h.add(new String("Writable")); h.add(new String("Executable")); h.add(new String("Readable")); h.add(new String("Writable")); h.add(new String("Executable")); System.out.println(h); // OUTPUT  }  }  Which of the following option gives the number of objects that will be eligible for garbage collection at the line commented //OUTPUT in the above program? | 3 objects |

|  |  |  |
| --- | --- | --- |
| CoreJava- GarbageCollection-018 | Consider the following code:  class SuperBase{ String string;  SuperBase(String string){ this.string = string;  }  public void setString(String string){ this.string = string;  }  }  public class UnReachability{  public static void main(String[] args){  UnReachability unreachability = new UnReachability(); unreachability.reach();  }  public void reach(){  SuperBase sb1 = new SuperBase("First"); // LINE 1 sb1.setString(""); // LINE 2  SuperBase sb2 = new Base("Second"); // LINE 3 sb1 = sb2; // LINE 4  }  }  Which of the following gives the correct place in the above code where the object referenced by sb1 becomes eligible for garbage collection? | At the line commented as  // LINE 4 |
| CoreJava- GarbageCollection-019 | Which of the following code snippets make objects eligible for Garbage Collection?   1. String s = "new string"; s = s.replace('e', '3'); 2. String s = "old string"; s = s.replace('6', 'r'); 3. StringBuffer sb = new StringBuffer("BufferedString"); String s = sb.toString(); 4. Serializable serializable = (Serializable) "Serializable object";   String s2 = serializable.toString(); serializable = null;   1. String replaceable = "replaceable"; StringBuffer sb = new StringBuffer(replaceable); replaceable = null;   sb = null; | 1,5 |

|  |  |  |
| --- | --- | --- |
| CoreJava-Inheritance- 001 | Consider the following code: interface Declare {  Declaration 1: protected int a = 5; Declaration 2: public static final int e = 9; Declaration 3: volatile int c = 7;  Declaration 4: transient int d = 8;  }  Which of the following option gives the declarations that results in compilation error? | Declaration 1,3,4 |
| CoreJava-Inheritance- 002 | Which of the following statement is true? | Has-a relationships always rely on instance variables. |
| CoreJava-Inheritance- 003 | Consider the following code:  Line No 1:public class CallMe { Line No 2: int i=10;  Line No 3: public void view(){ Line No 4: System.out.println(j); Line No 5: }  Line No 6: class WishMe{ Line No 7: int j=20;  Line No 8: public void view(){ Line No 9: System.out.println(i); Line No 10: }}}  Which of the following option gives the valid output for the above code? | Compilation error at line no:4 |

|  |  |  |
| --- | --- | --- |
| CoreJava-Inheritance- 004 | Consider the following code:  Line No 1:public class Calculate { Line No 2:int i;  Line No 3:void add()  Line No 4:{System.out.println("Addition Method returns nothing");}  Line No 5:int add()  Line No 6:{System.out.println("Addition Method returns integer");  Line No 7: return i;} Line No 8:}  Line No 9:public class TestCalculate {  Line No 10: public static void main(String args[]) Line No 11: {Calculate c=new Calculate();  Line No 12: c.add(); }}  Which of the following option gives the valid output for the above code? | compilation error at line no 5 |
| CoreJava-Inheritance- 005 | Consider the following code:  public class Welcome { String title;  int value;  public Welcome() { title += "JAVA";  }  public Welcome(int value) { this.value = value;  title = "hello"; Welcome();  }  public static void main(String args[]) { Welcome t = new Welcome(); System.out.println(t.title);  }}  Which of the following option gives the valid output for the above code? | Compilation fails |

|  |  |  |
| --- | --- | --- |
| CoreJava-Inheritance- 006 | Which of the following statements are true about has-a and is-a relationships?   1. Inheritance represents an is-a relationship 2. Inheritance represents a has-a relationship 3. Interfaces must be used when creating a has-a relationship 4. Instance variables can be used when creating a has-a relationship 5. Local variables must be used when creating has-a relationship | 1,4 |
| CoreJava-Inheritance- 007 | Consider the following code:  Line No:1. class Super {  Line No:2. public float getNum() { return 3.0f; } Line No:3. }  Line No:4. public class Sub extends Super { Line No:5//insert missing code here  Line No:6. }  Which of the following option gives the method, when replaced at line 6, causes unsuccessful compilation? | public void getNum() { } |
| CoreJava-Inheritance- 008 | Consider the following Statements:  Statement A: Anonymous inner class can be created in initializer or static blocks  Statement B: Anonymous inner class has no constructor  Which of the following option is true regarding the above given statements? | Both Statements A and B are true |

|  |  |  |
| --- | --- | --- |
| CoreJava-Inheritance- 009 | Consider the following code:  Line 1:class A {  Line 2: void display() { } Line 3:}  Line 4:class B extends A {  Line 5: // insert missing code here Line 6:}  Which of the following options give the code snippets, when inserted individually at the line no 5, will correctly complete the definition of class B?   1. int display() { /\* more code here \*/ } 2. void display() { /\* more code here \*/ } 3. private void display() { /\* more code here \*/ } 4. protected void display() { /\* more code here \*/ } | 2,4 |
| CoreJava-Inheritance- 010 | Consider the following code: class PriceList{  public float CalPrice( float num ) {  // assumed to have some processing code here  }  }  class Invoice extends PriceList{  public double CalPrice( double num ) {  // assumed to have some processing code here  }  }  Which of the following statement is true regarding the above code snippet? | It will compile but will not override the CalPrice method because of the different parameter list. |

|  |  |  |
| --- | --- | --- |
| CoreJava-Inheritance- 011 | Consider the following code:  interface Joy {String s1 = "Reading";}  class Human implements Joy {String s1 = "Books";} class Man extends Human {String s1 = "Writing";} public class Employee extends Man {  String s1 = "Books"; void printIt() {  System.out.print(((Human)this).s1 + ((Man)this).s1 + ((Employee)this).s1 + ((Joy)this).s1);  }  public static void main (String[] args) {new Employee().printIt();}  }  Which of the following option gives the valid output for the above code? | BooksWritingBooksReading |
| CoreJava-Inheritance- 012 | Consider the following code:  public class Base {  protected int count = 100; public int getCount() {  return count;  }  }  public class Detived extends Base { protected int count= 10;  public int getSize(){ return count;  }  public static void main(String[] args) { Base b = new Derived();  System.out.println (b.count + "," + b.getCount());  }  }  Which of the following option gives the valid output for the above code? | 100,10 |

|  |  |  |
| --- | --- | --- |
| CoreJava-Inheritance- 013 | Consider the following code:  interface Payroll { public void getSalary(); } abstract class Employee { public abstract void  getSalary(); }  Which of the following option gives the correct implementation that uses the Payroll interface and Employee class? | public class Manager extends Employee implements Inter {  public void getSalary() {  /\*do something\*/ }  } |
| CoreJava-Inheritance- 014 | Consider the following statement that represents a relationship:  “Shyam has a best friend who is a Tree”:  Which of the following option represents the above relationship correctly? | class Shyam { private Tree bestFriend; } |
| CoreJava-Inheritance- 015 | consider the following code:  Line No:1 public class MovieRelease Line No:2 {  Line No:3 public static void main(String[] args) { Line No:4 class Movie {  Line No:5 public String name; Line No:6 public Movie(String s) { Line No:7 name = s;  Line No:8}}  Line No:9 Object obj = new Movie("MaskOfZoro"); Line No:10 System.out.println(obj.name);  Line No:11} }  Which of the following option gives the valid output for the above code? | Zippo |

|  |  |  |
| --- | --- | --- |
| CoreJava-Inheritance- 016 | Consider the following Statements:  Statement A: Anonymous inner class can extend a class and implement an interface at the same time.  Statement B: Anonymous class can have their own members.  Which of the following option is true regarding the above statements? | Statement B is true and A is false |
| CoreJava-Inheritance- 017 | Consider the following code:  public class TestObj {  public static void main(String[] args) { Object o = new Object() {  public boolean equals(Object obj) { return TRUE;  }}  System.out.println(o.equals("Fred"));  }}  Which of the following option gives the valid output for the above code? | Compilation fails |
| CoreJava-Inheritance- 018 | Consider the following code:  public interface Runnable { void run(); }  Which of the following option gives the valid way of constructing an anonymous inner class instance? | Runnable r = new Runnable() { }; |
| CoreJava-Inheritance- 019 | Consider the following code:  Line no 1:class Outer {  Line no 2:public static class Inner { Line no 3:}  Line no 4:public static void display() { } } Line no 5:public class Test  Line no 6:{  Line no 7:public static void main(String args[]) Line no 8:{  Line no 9:// Replace with code from the option below Line no 10:}}  Which of the following option when replaced at line no 9, instantiates an instance of the nested class? | Outer.Inner o = new Outer.Inner(); |

|  |  |  |
| --- | --- | --- |
| CoreJava-Inheritance- 020 | Consider the following Code Snippet:  public class TestA  {  public int x; private int y; protected int z;  …  }  public class TestB extends TestA  {  protected int a; private int b;  …  }  public class TestC extends TestB  {  private int q;  …  }  Which of the following option gives the lists of instance data that are accessible in class TestB? | x, z, a, b |

|  |  |  |
| --- | --- | --- |
| CoreJava-Inheritance- 021 | Consider the following code:  abstract class drawing  {  final public void color(String s)  {  System.out.println(s);  }  abstract void draw();  }  class line extends drawing  {  void draw()  {  System.out.println("line");  }}  class circle extends line  {  void draw()  {  System.out.println("circle");  }}  public class Test  {  public static void main(String ar[])  { | blue line blue circle |
| CoreJava-Inheritance- 022 | Consider the following code:  Line no:1 interface iOne Line no:2 {int i=10; } Line no:3 interface iTwo Line no:4 { int i=10; }  Line no:5 public class TestInterface implements iOne,iTwo{  Line no:6 public static void main(String[] a) Line no:7 { System.out.println(i); }}  Which of the following option gives the valid output for the above code? | Compilation error at line no 7 |

|  |  |  |
| --- | --- | --- |
| CoreJava-Inheritance- 023 | Consider the following interface declarations:  interface A {void main(String[] args);} interface B {public void main(String[] args);}  interface C {public static void main(String[] args);} interface D {protected void main(String[] args);} interface E {private void main(String[] args);}  Which of the following option gives the valid interface declaration that will compile successfully? | interface A,B |
| CoreJava-Inheritance- 024 | Consider the following code:  interface A {int i = 1; int calculate();}  interface B extends A {int i = 10; int calculate();} class Check implements B {  public int calculate() {return ++i;} public static void main(String[] args) {  System.out.print(new Check().calculate());  }}  Which of the following option gives the valid output for the above code? | Compilation Error |
| CoreJava-Inheritance- 025 | Consider the following code:  Line No 1:class Test {  Line No 2:public static void main (String[] args) { Line No 3:byte b = 1;  Line No 4:long lg = 1000; Line No 5:b += lg;  Line No 6: }}  Which of the following option gives the valid output for the above code? | Code compiles successfully without output |

|  |  |  |
| --- | --- | --- |
| CoreJava-Inheritance- 026 | Consider the following code:  interface First {} interface Second {} class Base implements First {}  class Derived extends Base implements Second {} class Test {  public static void main(String args[]) { Line No 1: Derived s1 = new Derived(); Line No 2: Second i2 = s1;  Line No 3: First i1 = s1; Line No 4: Base base = s1;  Line No 5: Derived s2 = (Derived)base;  }}  Which of the following option gives the valid output for the above code? | x, y, z, a, b |
| CoreJava-Inheritance- 027 | Consider the following code:  class Assign {String s1 = "abc"; String s2 = "xyz";} class Print extends Assign {  String s1 = "ijk";  public static void main(String args[]) { Print x = new Print(); Assign y = (Assign)x;  System.out.println(x.s1+" "+x.s2+" "+y.s1+" "+y.s2);  }}  Which of the following option gives the valid output for the above code? | ijk xyz abc xyz |

|  |  |  |
| --- | --- | --- |
| CoreJava-Inheritance- 028 | Consider the following code:  Line no 1:class Top {  Line no 2:private int i1, i2;  Line no 3:void print() {System.out.print("Top, i1="+i1+", i2="+i2);}  Line no 4:Top(int i1, int i2) {this.i1=i1; this.i2=i2;} Line no 5:}  Line no 6:class Check extends Top { Line no 7:private int i1, i2;  Line no 8:void print() {System.out.print("Check, i1="+i1+", i2="+i2);}  Line no 9:Check(int i1, int i2) {this.i1=i1; this.i2=i2;} Line no 10:public static void main(String[] args) { Line no 11: Top t = new Check(1,2); t.print();  Line no 12:}}  Which of the following option gives the valid output for the above code? | Compile time error at line no 9 |
| CoreJava-Inheritance- 029 | Consider the following code:  public class TestParent {  public static void main(String s[]) { Child c = new Child();  c.print();  Parent p = new Parent(); p.print();  p = c;  p.print(); // Line no 1  } }  class Parent { static int x = 100;  public static void print() { System.out.println(x); }  }  class Child extends Parent { static int x = 200;  public static void print() { System.out.println(x); }  }  Which of the following option gives the valid output for the above code? | 200100100 |

|  |  |  |
| --- | --- | --- |
| CoreJava-Inheritance- 030 | Consider the following code:  public class Test {  public static void main(String args[]) { A ref1 = new C(); System.out.println(ref1.f());  }}  class A {  private int f() { return 0; } public int g() { return 3; }  }  class B extends A { public int f() { return 1; }  public int g() { return f(); }  }  class C extends B { public int f() { return 2; }  }  Which of the following option gives the valid output for the above code? | Compilation Error |
| CoreJava-Inheritance- 031 | Consider the following code:  interface dumb {} interface Silent {} class Base implements dumb {}  class Derived extends Base implements Silent {} public class Test {  public static void main(String []args) { Base[] base = {new Base()}; // Line no 1  Derived dev[] = {new Derived()}; // Line no 2 Object obj = dev; // Line no 3  base = obj; // Line no 4  }}  At the time of compilation the above mentioned code generates some error.  Which of the following option gives the line no where the error is generated? | Line no 4 |

|  |  |  |
| --- | --- | --- |
| CoreJava-Inheritance- 032 | Consider the following code:  class First { static void print() {System.out.print("First");}} class Second extends First {  void print() {System.out.print("Second");}//Line no 1 static void print(String s) {System.out.print(s+",");}  }  class Test {  public static void main (String[] args)  {Second.print("main"); new Second().print();}//Line no 2  }  Which of the following option gives the valid output for the above code? | Compile time error at line no 1 |
| CoreJava-Inheritance- 033 | Consider the following code:  class Flower {  static void print(){System.out.print("Jasmine");} void printOnce() {System.out.print("Lilly");} void printAgain(){print();printOnce();}  }  public class Bouquet extends Flower {  static void print(){System.out.print("Rose");} void printOnce(){System.out.print("Sunflower");} public static void main(String[] args) {  new Bouquet().printAgain();  }}  Which of the following option gives the valid output for the above code? | JasmineSunflower |

|  |  |  |
| --- | --- | --- |
| CoreJava-Inheritance- 034 | Consider the following code:  public class BottomTest {  public static void main(String s[]) {  new OuterLevel().new Inner().new DeepInner().new DeepestInner().print();  }  }  class OuterLevel { String name = "Apple";  class Inner {  String name = "Mango"; class DeepInner {  class DeepestInner {  public void print() {  //Line no 1  }  } }}}  Which among the following statement when substituted to the line commented as // Line no 1 will make the | System.out.println(Inner.thi s.name); |

|  |  |  |
| --- | --- | --- |
| CoreJava-Inheritance- 035 | Consider the following code:  import java.io.\*; class GetData{  public void input()throws FileNotFoundException{}  }  public class PrintData extends GetData{ public static void main(String argv[]){ PrintData e = new PrintData();  }  public void input(){} protected PrintData(){  try{  DataInputStream din = new DataInputStream(System.in);  System.out.println("Welcome"); din.readByte(); System.out.println("Thankyou"); this.input();  }catch(IOException ioe) {}  }  }  Which of the following statement is true regarding the | Compile and run with output of "Welcome" and "Thankyou" after a key is hit |
| CoreJava-Inheritance- 036 | Consider the following code:  class Smile { protected joy h;  }  class Happy extends Smile { } class joy { }  Which of the following statement is correct regarding the above given code? | Happy has-a joy and Happy is-a Smile |

|  |  |  |
| --- | --- | --- |
| CoreJava-Inheritance- 037 | Consider the following code:  public class SuperTest { static int y = 2;  SuperTest(int x) { this(); y = y \* 2; } SuperTest() { y++; }  }  class Test extends SuperTest  {  Test() { super(y); y = y + 3; }  public static void main(String [] args)  {  new Test(); System.out.println(y);  }  }  Which of the following option gives the valid output for the above code? | 9 |
| CoreJava-Introduction- 001 | Which of the following options give the valid argument types for main() method?   1. String[] args 2. String args[] 3. String ..args 4. String args 5. String[] args[] | 1,2,3 |
| CoreJava-Introduction- 002 | Which of the following methods are not the member of Object class?   1. getClass() 2. run() 3. hashCode() 4. wait() 5. currentTimeMillis() | 2,5 |
| CoreJava-Introduction- 003 | Which of the following options give the member methods of Object class, that cannot be overriden?   1. equals() 2. hashCode() 3. wait() 4. notify() 5. clone() | 3,4 |

|  |  |  |
| --- | --- | --- |
| CoreJava-Introduction- 004 | Which of the following options give the valid package names?  1) dollorpack.$pack.$$pack 2) $$.$$.$$   1. \_score.pack. pack 2. [p@ckage.subp@ckage.innerp@ckage](mailto:p@ckage.subp@ckage.innerp@ckage) 3. .package.subpackage.innerpackage | 1,2,3 |
| CoreJava-Introduction- 005 | The term 'Java Platform' refers to . | Java Runtime Environment (JRE) |
| CoreJava-Introduction- 006 | Which of the following statement gives the use of CLASSPATH? | Holds the location of User Defined classes, packages and JARs |
| CoreJava-Introduction- 007 | Consider the following:  Assume that there is a package structure as follows: com.testpack  which contains a class called TestPack, with some static and non-static methods and a static inner class.  Which of the following options give the valid import statements for the above package structure and class, in order to import the class TestPack or its members?   1. import com.testpack.TestPack; 2. import com.testpack; 3. import com.testpack.TestPack.\*; 4. import static com.testpack.TestPack; 5. import static com.qb2020.TestPack.\*; | 1,3,5 |

|  |  |  |
| --- | --- | --- |
| CoreJava-Introduction- 008 | Which of the following are true about packages?   1. Packages can contain only Java Source files 2. Packages can contain both Classes and Interfaces (Compiled Classes) 3. Packages can contain non-java elements such as images, xml files etc. 4. Sub packages should be declared as private in order to deny importing them 5. Class and Interfaces in the sub packages will be automatically available to the outer packages without using import statement. | 2,3 |
| CoreJava-Introduction- 009 | Consider the following:  There is a package structure com.runpack contains the class Hello.class (contians main method)  The above package is stored under the folder 'C:\packtest'  Which of the following options give valid ways to execute the Hello.class file?   1. java -classpath C:\packtest com.runpack.Hello 2. java com.runpack.Hello keeping 'C:\packtest' as current directory 3. java com/runpack/Hello keeping 'C:\packtest' as current directory 4. java Hello keeping 'C:\packtest\com\runpack' as current directory 5. java com\runpack\Hello keeping 'C:\packtest' as current directory | 1,2,3 |
| CoreJava-Introduction- 010 | Which of the following following option gives the name of Project property in SDE that helps to refer to the third party JAR libraries? | Java Build Path |
| CoreJava-Introduction- 011 | Which of the following options give the commands that are provided in the 'Source' menu of SDE?   1. Format 2. Correct Indentation 3. Generate Getters and Setters 4. Rename 5. Move | 1,2,3 |

|  |  |  |
| --- | --- | --- |
| CoreJava-Introduction- 012 | Which of the following option gives one possible use of the statement 'the name of the public class should match with its file name'? | Helps the compiler to find the source file that corresponds to a class, when it does not find a class file while compiling |
| CoreJava-Introduction- 013 | Consider the following code:  public class TestMain {  public static void main(String[] args) { System.out.println("TestMain"); TestMyClass.main(new String[] {"one", "two"});  }  }  public class TestMyClass {  public static void main(String[] args) { for(String s:args) System.out.println(s);  }  }  Which of the following statement is true regarding the above code? | On compiling and running the class TestMain, it prints TestMain  one two |
| CoreJava-Introduction- 014 | Which of the following options give the Characteristics of Java, that made Java popular in Web Development?   1. Object Oriented 2. Interpreted 3. Robust 4. Portable 5. Secure | 4,5 |
| CoreJava-Introduction- 015 | Which of the following statement is true? | Classes can be loaded at Runtime, without actually referring the class in the code at compile time. |

|  |  |  |
| --- | --- | --- |
| CoreJava-Introduction- 016 | Which of the following option gives the correct sequence in which the Java Virtual Machine searches for a class? | Java Class Library Packages in the extension directory of JRE / JDK  User-defined packages and libraries |
| CoreJava-Introduction- 017 | Which of the following statements are true regarding java.lang.Object class?   1. Object class is an abstract class 2. Object class cannot be instantiated directly 3. Object class has the core methods for thread synchronization 4. Object class provides the method for Set implementation in Collection framework 5. Object class implements Serializable interface internally | 3,4 |
| CoreJava-IO-001 | Which of the following listed code snippets create a new file?   1. File f = new File("WELCOME.TXT"); 2. new File("WELCOME.TXT").createNewFile(); 3. FileWriter fw = new FileWriter("WELCOME.TXT"); 4. BufferedReader br = new BufferedReader(new InputStreamReader(new FileInputStream("WELCOME.TXT"))); 5. FileReader fr = new FileReader("WELCOME.TXT"); | 2,3 |

|  |  |  |
| --- | --- | --- |
| CoreJava-IO-002 | Consider the following scenario:  An image processing project is being developed using Java.  The file extensions of the types of image files that can be processed are stored in a TreeSet.  The List of files with the valid file types need to be listed in custom dialog box.  Which of the following options give the possible classes / interfaces, that can be used, in order to achieve the above mentioned functionality?   1. File class 2. FilenameFilter interface 3. FileFilter interface 4. FileReader class 5. FileInputStream class | 1,2,3 |
| CoreJava-IO-003 | Consider the following scenario:  A file manager application is being developed in Java.  One important feature of this application is showing the properties such as file size, last modified date, file permissions etc., of a selected file or directory.  Which of the following option gives the possible classes / interfaces, that can be used to achieve the above mentioned functionality? | File class |

|  |  |  |
| --- | --- | --- |
| CoreJava-IO-004 | Consider the following scenario:  A file search utility need to be developed using Java.  The utility need to search on a specified drive and/or directory. Search should recurse into the sub-directories. Search criteria would be the name of the file with wild cards, last modified date or range of dates, file/directory attributes such as hidden, read-only, file size etc.,  Which of the following options give the possible classes / interfaces, that can be used to achieve the above mentioned functionality?   1. File class 2. FilenameFilter interface 3. FileSearchFilter interface 4. FileInputStream class 5. FileReader class | 1,2 |
| CoreJava-IO-005 | Consider the following scenario:  A file search utility which is already developed in Java needs to be enhanced with the additional feature.  Currently the utility searches the file system with various search criteria such as filename with wild cards, file size, last modified date and other attributes like hidden, ready- only and file size etc.  Now as an additional feature, the content of the file also need to be scanned during file search, to make the search even more intelligent.  Which of the following options give the possible classes / interfaces, that can be used to achieve the above mentioned functionality?   1. File class 2. FileInputStream class 3. FileReader class 4. InputStreamReader class 5. BufferedReader | 2,4,5 |

|  |  |  |
| --- | --- | --- |
| CoreJava-IO-006 | Consider the following scenario:  An integrated text editor needs to implemented as a part of software which is being developed in Java.  The text files need to be read line by line from the file. In the same, the lines need to be written line by line to the file.  Which of the following options give the possible classes / interfaces, that can be used to achieve the above mentioned functionality?   1. LineInputStream and LineOutputStream 2. BufferedReader and BufferedWriter 3. FileReader and FileWriter 4. TextReader and TextWriter 5. FileInputStream and FileOutputStream | 2,3 |
| CoreJava-IO-007 | Consider the following scenario:  A file contains a 10 x 10 matrix containing 100 double values. All the 100 values are stored in sequential manner.  The row and column are taken as input from the keyboard and the corresponding value for the given row and the column has to be looked up from the matrix file.  Which of the following option gives the possible classes / interfaces, that can be used, in order to achieve the above mentioned functionality? | RandomAccessFile |

|  |  |  |
| --- | --- | --- |
| CoreJava-IO-008 | Consider the following scenario:  A shopping cart application is being developed in Java Enterprise Technology, for a popular shopping mall in the city.  Following are identified Classes that need to be used in the software system.  Customer - stores the customer information Product - stores the product information  UserInfo - stores the information such as username, date and time of login for currently logged in user  CartItem - stores the information about the product selected by the customer and its quantity  Cart - stores the collection of CartItems and total amount Invoice - stores the information about the billed items, including customer information, quantity, rate and total amount.  CardInfo - stores the customer's credit card number, CVV number, card expiry month and year.  Which of the following option gives the name of the class types that would not require Serializable? | CardInfo |

|  |  |  |
| --- | --- | --- |
| CoreJava-IO-009 | Consider the following scenario:  A mobile service provider has a large customer base. Our company has taken the project of building their ERP system using Java Technology. There are various modules under the ERP. Some of the noteable moudles are CRM, Online Bill Presentment and Payment, Billing and Accounting. Below are some of the tasks that come under these modules.  Task I. As a part of online bill presentment, the data in the Bill Object needs to be generated as PDF document Task II. As a part of online bill payment module, the information submitted by the user need to be submitted to the server in a secure way.  Task III. The same Customer Object used for showing the customer details in the Online Bill Presentment system, should be used in CRM module also.  Task IV. As a part of payment module, a payment confirmation Text Message needs to be sent to the Customer's mobile phone  Task V. As a part of Billing and Accounting module, a bill need to be generated on the billing date for every customer  Which of the following option gives the task that involves Serialization process? | Task III |
| CoreJava-IO-010 | Consider the following scenario:  A simple type of file encryption is required to implement in Java.  The content of a file has to be considered as possible two equals parts and the alternative bytes of those two parts has to be swapped.  Which of the following option gives the possible classes / interfaces, that can be used to achieve the above mentioned functionality? | RandomAccessFile |

|  |  |  |
| --- | --- | --- |
| CoreJava-IO-011 | Consider the following code:  import java.io.BufferedInputStream; import java.io.BufferedReader; import java.io.FileInputStream; import java.io.FileReader;  import java.io.IOException;  import java.io.InputStreamReader;  public class StreamTest2 {  public static void main(String[] args) throws IOException {  String filename = "TestData.txt";  /\* CODE \*/  String line = null; do {  line = pipe3.readLine(); if(line != null)  System.out.println(line);  } while(line != null); pipe3.close();  }  }  The above program needs to read the text file | 1,2,4 |

|  |  |  |
| --- | --- | --- |
| CoreJava-IO-012 | Consider the following code:  import java.io.Externalizable; import java.io.FileInputStream; import java.io.FileOutputStream; import java.io.IOException; import java.io.ObjectInput; import java.io.ObjectInputStream; import java.io.ObjectOutput;  import java.io.ObjectOutputStream; import java.io.Serializable;  class Employee implements Externalizable { public Integer id;  public String name; public Double salary;  public Employee() { }  public Employee(Integer id, String name, Double salary)  { this.id = id; this.name = name; this.salary = salary; } public void readExternal(ObjectInput in) throws  IOException, ClassNotFoundException { this.id = Integer.valueOf(in.readLine()); this.name = in.readLine();  this.salary = Double.valueOf(in.readLine());  }  public void writeExternal(ObjectOutput out) throws | 2,5 |

|  |  |  |
| --- | --- | --- |
| CoreJava-IO-013 | Consider the following code: import java.io.Serializable;  class CustomerInfo implements Serializable { public String customerId;  public String customerName; public transient String cardNo; public transient String cvvNo;  }  When the object of the above class is serialized, all the values of the object except cardNo and cvvNo will be serialized.  Which of the following code gives the alternative way of getting the same functionality as the above code, without using the transient keyword? | import java.io.Serializable;  class CardInfo {  public String cardNo; public String cvvNo;  }  class CustomerInfo extends CardInfo implements Serializable {  public String customerId; public String  customerName;  } |

|  |  |  |
| --- | --- | --- |
| CoreJava-IO-014 | Consider the following code:  import java.io.FileInputStream; import java.io.IOException; import java.io.RandomAccessFile;  public class TestRandom1 {  public static void main(String[] args) throws IOException {  FileInputStream fin = new FileInputStream("C:/TestRandom.txt");  RandomAccessFile raf = new RandomAccessFile("C:/TestRandom.txt", "a");  raf.seek(fin.available()); fin.close();  raf.writeBytes("Next Info"); raf.close();  }  }  Which of the following statement is true regarding the above code? | Runtime Error 'IllegalArgumentException: invalid file open mode' |

|  |  |  |
| --- | --- | --- |
| CoreJava-IO-015 | Consider the following code:  import java.io.FileInputStream; import java.io.IOException; import java.io.RandomAccessFile;  public class TestRandom2 {  public static void main(String[] args) throws IOException {  FileInputStream fin = new FileInputStream("C:/TestRandom.txt");  RandomAccessFile raf = new RandomAccessFile("C:/TestRandom.txt", "r");  byte data[] = new byte[20];  raf.seek(fin.available()); fin.close();  raf.seek(-10);  int current = (int) raf.getFilePointer(); raf.read(data, current, 7); System.out.println(new String(data)); raf.close();  }  } | Runtime error 'Negative seek offset' |

|  |  |  |
| --- | --- | --- |
| CoreJava-IO-016 | Consider the following code:  import java.io.IOException; import java.io.RandomAccessFile;  public class TestRandom {  public static void main(String[] args) throws IOException {  RandomAccessFile raf = new RandomAccessFile("C:/TestRandom.txt", "rw");  /\* CODE \*/ raf.close();  }  }  Which of the following code snippets when replaced exclusively for the comment /\* CODE \*/ in the above code, will set the file size to 1 Kilo Bytes?  1) raf.seek(1024);  2) raf.seek(1023); raf.writeBytes("A");  3) raf.setLength(1024); | 2,3 |

|  |  |  |
| --- | --- | --- |
| CoreJava-IO-017 | Consider the following code:  import java.io.Serializable; import java.io.FileInputStream; import java.io.FileOutputStream; import java.io.IOException; import java.io.ObjectInput;  import java.io.ObjectInputStream; import java.io.ObjectOutputStream;  interface CustomSerializable extends Serializable { }  class CustomSerializableClass implements CustomSerializable {  private String data;  public String getData() { return data; }  public void setData(String data) { this.data = data; }  }  public class TestCustomSerializable { public static void main(String[] args)  throws IOException, ClassNotFoundException { CustomSerializableClass css = new  CustomSerializableClass(); css.setData("Sample Data"); ObjectOutputStream objO = new  ObjectOutputStream( | No errors in the program. Prints 'Sample Data' |

|  |  |  |
| --- | --- | --- |
| CoreJava-IO-018 | Consider the following code:   1. import java.io.File; 2. import java.io.FileFilter; 3. import java.io.FilenameFilter; 04 4. public class ListReadOnlyFiles { 5. public static void main(String[] args) { 6. File file = new File("D:/Documents"); 08 7. File[] fileList = file.listFiles(new ReadOnlyFilter()); 8. for (File f : fileList) { 9. System.out.println(f.getName()); 12 }   13 }  14 }  The above written code is intended to list only the read- only files from the folder 'D:/Documents'. The code is incomplete and any one of the listed class implementation need to be instantiated and passed as parameter to the listFiles() method at line number 09 in the above code.  Which of the following option gives the correct version of class implementation to make the above code achieve the required functionality? | class ReadOnlyFilter implements FileFilter {  public boolean accept(File file) {  if(file.canRead() &&  !file.canWrite())  return true; else  return false;  }  } |

|  |  |  |
| --- | --- | --- |
| CoreJava-IO-019 | Consider the following code:   1. import java.io.File; 2. import java.io.FileFilter; 3. import java.io.FilenameFilter; 04 4. public class ListFiles { 5. public static void main(String[] args) { 6. File file = new File("D:/Personal"); 7. Filter filter = new CustomFilter(); 09   10 File[] fileList = file.listFiles(filter); 11   1. for (File f : fileList) { 2. System.out.println(f.getName()); 14 }   15 }  16 }  Following is the filter implementation used at line number 08 in the above program:  interface Filter extends FilenameFilter, FileFilter { } class CustomFilter implements Filter {  public boolean accept(File file, String name) {  if(name.startsWith("A")) | Compile time error at line number 10 |

|  |  |  |
| --- | --- | --- |
| CoreJava-IO-020 | Consider the following code:  import java.io.BufferedWriter; import java.io.FileWriter; import java.io.IOException;  public class StreamTest1 {  public static void main(String[] args) throws IOException {  String msg[] = {  "Unable to deliver your mail", "Message Sent",  "Invalid URL", "Invalid File Format", "Protocol Error", "File Not Found"  };  BufferedWriter bw = new BufferedWriter( new FileWriter("C:/TestData.txt"));  for(String s : msg) { bw.write(s);  }  }  }  Which of the following statement is true regarding the | No errors in the program. Compiles and executes successfully.  But creates an empty file 'TestData.txt' |
| CoreJava-JDBC-001 | Which of the following types of drivers are not vendor specific implementation of a Java driver?   1. Type 1 driver 2. Type 2 driver 3. Type 3 driver 4. Type 4 driver | 1,3 |
| CoreJava-JDBC-002 | driver follows a three-tiered architecture. | Type 3 |
| CoreJava-JDBC-003 | Which of the following driver will be most suitable for those Java Client Applications that run on different operating systems but need to connect to a centralized database server? | Type 4 driver |
| CoreJava-JDBC-004 | Which of the following option gives the return value of getUpdateCount(), when the last result is a ResultSet? | -1 |

|  |  |  |
| --- | --- | --- |
| CoreJava-JDBC-005 | Which of the following options give new features added in JDBC 2.0?   1. DatabaseMetaData 2. Scrollable ResultSets 3. ResultSetMetaData 4. Batch Updates 5. Programmatic inserts, deletes and updates | 2,4,5 |
| CoreJava-JDBC-006 | Which of the following option gives the valid method to get the ResultSetMetaData from a ResultSet object? | getMetaData() |
| CoreJava-JDBC-007 | Which of the following option gives the valid method to get the DatabaseMetaData from a Connection object? | getMetaData() |
| CoreJava-JDBC-008 | Which of the following listed Class / Interface acutally establishes the connection to the database using the given Connection URL String? | Driver |
| CoreJava-JDBC-009 | Which of the following listed option gives the valid type of object to store a date and time combination using JDBC API? | java.sql.Timestamp |
| CoreJava-JDBC-010 | Which of the following options give the valid methods that can be used for executing DDL statements?   1. executeQuery() 2. executeUpdate() 3. execute() 4. executeDDL() 5. executeResult() | 2,3 |
| CoreJava-JDBC-011 | The getPrimaryKeys() method is in which of the following listed interface? | DatabaseMetaData |
| CoreJava-JDBC-012 | Which of following statement is true regarding parameter usage in CallableStatement? | All OUT parameteres must be registered with CallableStatement object prior to the execution |
| CoreJava-JDBC-013 | Which of the following option gives the default Concurrency of a ResultSet? | CONCUR\_READ\_ONLY |

|  |  |  |
| --- | --- | --- |
| CoreJava-JDBC-014 | Which of the following option gives the valid way to pass the values, while inserting a row using ResultSet? | The updater methods are used to set the values for the new row |
| CoreJava-JDBC-015 | Which of the following options are valid sub classes of java.util.Date class?   1. java.sql.Time 2. java.sql.ShortDate 3. java.sql.DateTime 4. java.sql.Timestamp 5. java.sql.LongDate | 1,4 |
| CoreJava-JDBC-016 | Which of the following option is a valid interface that gives the information about the Tables, Views, Stored Procedures and Other database objects in a Database? | DatabaseMetaData |
| CoreJava-JDBC-017 | Which of the following statement is tue for an active Connection object for which the auto-commit feature is set to true? | Starts a new Transaction for every SQL statement that are executed under that connection. |

|  |  |  |
| --- | --- | --- |
| CoreJava-JDBC-018 | Consider the following code snippet:  static { try {  Class.forName("oracle.jdbc.OracleDriver");  } catch(ClassNotFoundException cnfe) { System.out.println("Driver not found");  }  }  Which of the following statements are true regarding the above code snippet?   1. It just loads the OracleDriver class, instantiation happens at the time of connection 2. It loads the OracleDriver class, and instantiates a DriverManager object and returns it 3. It loads the OracleDriver class, instantiates it and registers it with DriverManager class 4. oracle.jdbc.OracleDriver is a DriverManager class type object 5. oracle.jdbc.OracleDriver is a Driver interface type object | 3,5 |
| CoreJava-JDBC-019 | Consider the following code snippet:   1. PreparedStatement s = c.prepareStatement("create table a(b int)"); 2. s.executeUpdate(); 3. s.close();   Assume that c is a valid Connection object, which is properly connected to a database.  Which of the following statement is true regarding the above code snippet? | No problem in the code. Executes without any exception and creates the table 'a' |
| CoreJava-JDBC-020 | Which of the following statement is true regarding isSearchable(int column) method of ResultSetMetaData interface? | Returns true if the column is of any primitive type, and false if the column type is BLOB, Image. |

|  |  |  |
| --- | --- | --- |
| CoreJava-JDBC-021 | Which of the following option gives the valid way to check from a Java application, whether a table exists or not? | getTables() method in DatabaseMetaData can be used to query the available table names |
| CoreJava-JDBC-022 | Which of following statements are true regarding CallableStatement?   1. A CallableStatement can return Java Primitive data types 2. A CallableStatement can return SQL data types defined in java.sql.Types 3. A CallableStatement can return one ResultSet object for a single call 4. A CallableStatement can return multiple ResultSet objects for a single call 5. A CallableStatement can return Java Wrapper types | 2,3,4 |
| CoreJava-JDBC-023 | Which of the following will be the output when a getter method is called with a column name and the ResultSet has several columns with the same name? | The value of the first matching column will be returned |
| CoreJava-JDBC-024 | Which of the following option gives the status of a ResultSet object, when the Statement object that generated it is re-executed? | The ResultSet object is automatically closed |

|  |  |  |
| --- | --- | --- |
| CoreJava-JDBC-025 | Consider the following code snippet:   1. String sql = "select sample\_pk, sample\_data from sample3" 2. PreparedStatement ps = c.prepareStatement(sql); 3. ResultSet rs = ps.executeQuery();   Assume that c is a valid Connection object, which is already connected to the database. And there is no table with the name 'sample3' in the database.  Which of the following statement is true about the above code? | SQLException is thrown at the line number 3 |
| CoreJava-JDBC-026 | Which of the following statements give the various points at which the commit occurs, for an active Connection object for which the auto-commit feature is set to true?   1. When a statement with INSERT, UPDATE and DELETE query completes 2. When a statement with SELECT query returns the ResultSet 3. When the last row of the ResultSet is retrieved 4. When the ResultSet object is closed 5. When the connection is closed | 1,3,4 |
| CoreJava-Keywords- 001 | Consider the following code:  public class Main {  public String value="Face value"; public Main() {  value="Main value"; System.out.println(value);  }  {  System.out.println(value); value="System value"; System.out.println(value);  }  public static void main(String[] args) { Main n=new Main();  }  }  Which of the following option gives the output for the above code? | Face value System value Main value |

|  |  |  |
| --- | --- | --- |
| CoreJava-Keywords- 002 | Consider the following code snippet:   1. long test = 045; 2. float f = -132; 3. int value = (int)true;   4) double d = 0x12345678;  5) short s = 20;  Which of the following option gives the legal assignments? | 1 2 4 5 |
| CoreJava-Keywords- 003 | Consider the following statements:   1. Every floating-point literal is implicitly a double, not a float. 2. In the declaration byte b=120; int literal 120 is implicitly converted to byte.   Which of the following option is valid regarding the above statements? | Both A and B are true |
| CoreJava-Keywords- 004 | Consider the following code:  class CastingTest {  public static void main(String [] args) { long l = 130L;  byte b = (byte)l;  System.out.println("The Value of byte b is: " + b);  }  }  Which of the following option gives the output for the above code? | Prints the output "The Value of byte b is:-126 |
| CoreJava-Keywords- 005 | Consider the following code:  public class FetchName {  public static void main(String[] args) {  char[] pickFrom = { 'a', 'b', 'c', 'd', 'e', 'f', 'g',  'h', 'i', 'j', 'k', 'l', 'm' };  char[] dumpTo = new char[7]; System.arraycopy(pickFrom, 2, dumpTo, 0, 7); System.out.println(new String(dumpTo));  }  }  Which of the following option gives the output for the above code? | cdefghi |

|  |  |  |
| --- | --- | --- |
| CoreJava-Keywords- 006 | Consider the following code snippet:   1. public class Arrays1 { 2. public static void main(String args[]) { 3. String[] s;   4 s = {"abc","def","hij");   1. for(int val=0; val<= s.length; val++) { 2. System.out.println(s[val]); 3. System.out.println(","); 8 }   9 int[] arr = new int[]{1, 2, 3};   1. for (int count = 0; count <= arr.length; count++) { 2. System.out.println(arr[count]); 3. System.out.println(","); 13 }   14 }  15 }  16 }  Which of the following statement is true regarding the above code? | The above code generates a Compilation Error |
| CoreJava-Keywords- 007 | Consider the following Statements:  Statement A: All enums are subclasses of interface java.lang.Enum.  Statement B: All instances of Enums are serializable by default.  Which one of the following option is valid for the above statements? | Statement A is false and B is true |

|  |  |  |
| --- | --- | --- |
| CoreJava-Keywords- 008 | Consider the following code snippet:  import org.junit.\*;  import static org.junit.Assert.\*; import java.io.\*;  public class OutputTest { private File output;  @Before  public void createOutputFile() { output = new File(...);  }  @After  public void deleteOutputFile() { output.delete();  }  @Test  public void testSomethingWithFile() {  ...  }  }  Which of the following option gives the order in which | createOutputFile() testSomethingWithFile() deleteOutputFile() |
| CoreJava-Keywords- 009 | Consider the following code:  public class question1 {  public static void main(String args[])  {  Boolean b1 = new Boolean("TRUE"); Boolean b2 = new Boolean("true"); Boolean b3 = new Boolean("JUNK"); System.out.println( b1 + b2 + b3);  }}  Which of the following option gives the output for the above code? | Compile Time error |

|  |  |  |
| --- | --- | --- |
| CoreJava-Keywords- 010 | Consider the following code:  public class PredictName {  public static void main(String [] args) { String name = "unknown";  int shirtSize = 42; char sex = 'm';  double salary = 10000;  if (shirtSize >=40) name = "Jimmy";  if (salary > 16121) name = "Cameron " + name; if (sex >= 'f') name = "female " + name;  System.out.println("The Person Name is " + name);  }  }  Which of the following option gives the output for the above code? | The Person Name is female Jimmy |
| CoreJava-Keywords- 011 | Consider the following Statements about Operators:   1. The instanceof operator can not be applied to an array. 2. The instanceof operator can be applied to object reference. 3. The equals method compares content of two objects 4. The == operator tests reference of two objects   Which of the following option is valid regarding the above statements? | B,C,D is true and A is false |
| CoreJava-Keywords- 012 | Consider the following code:  Line No 1:public class JavaByteDataType {  Line No 2: public static void main(String args[]) Line No 3:{  Line No 4: byte a,b,c; Line No 5: a=1;  Line No 6: b=1; Line No 7: c=a+b;  Line No 8: System.out.println("value of c is"+c); Line No 9: }  Which of the following option gives the output for the above code? | Compile time error at line number 7 |

|  |  |  |
| --- | --- | --- |
| CoreJava-Keywords- 013 | Consider the following code:  Line no 1:public class ValidDeclaration{  Line no 2: public static void main(String args[]) Line no 3: {  Line no 4: byte a=126; Line no 5: byte b=127; Line no 6: byte c=129;  Line no 7: int e=3333333333; Line no 8: }  Line no 9:}  Also consider the following statments regarding the above code:   1. All the above declarations are correct 2. Line 4 should be changed to byte b=(byte)127; 3. Line 5 should be changed to byte c=(byte)129; 4. Line 6 should be changed to int e=(int)3333333333;   Which of the following option is valid regarding the above code and statements? | Statement C and D are correct |
| CoreJava-Keywords- 014 | Consider the following code:  1 public class Array2 2 {   1. public static void main(String args[]){ 2. int[] myA; 3. myA = new int[2]; 4. myA[] myB; 5. myB = new myA[3]; 8 myB[0]=a;   9 myB[1]=1;  10 myB[2]=3;   1. System.out.println(myA[0]); 2. System.out.println(myB[1]); 3. System.out.println(myB[0]);   14 }}  Which of the following option gives the output for the above code? | Compile time error at line no 7 |

|  |  |  |
| --- | --- | --- |
| CoreJava-Keywords- 015 | Consider the following code:  public class DisplayNames {  public static void main(String[] args) { String[][] names = {{"Mr. ", "Mrs. ", "Ms. "},  {"Smith", "Jones"}}; System.out.println(names[0][0] + names[1][1]); System.out.println(names[0][2] + names[1][0]);  }}  Which of the following option gives the output for the above code? | Mr.Jones Ms.Smith |
| CoreJava-Keywords- 016 | Consider the following code:  Line NO:1 public enum Flower { Line NO:2 Rose ("Pink"),  Line NO:3 Lotus("White"), Line NO:4 Lilly ("Violet"); Line NO:5 String colour;  Line NO:6 Flower(String colour) Line NO:7 {  Line NO:8 this.colour=colour; Line NO:9 }  Line NO:10 public static void main(String args[]) Line NO:11 {  Line NO:12 System.out.println(Flower.Rose); Line NO:13 System.out.println(Lotus);  Line NO:14 System.out.println(Lilly); Line NO:15 }  Line NO:16 }  Which of the following option gives the output for the above code? | Rose Lotus Lilly |

|  |  |  |
| --- | --- | --- |
| CoreJava-Keywords- 017 | Consider the following code:  public enum Greeting {  Morning("GoodMorning"){public String toString(){return "KeepSmiling!..";}},  Noon("GoodAfteernoon"), Evening("GoodEvening"), Night("GoodNight"); String message; Greeting(String message)  {  this.message=message;  }  public static void main(String args[])  {  for(Greeting g:Greeting.values())  {  System.out.println(g);  }  } }  Which of the following option gives the output for the above code? | KeepSmiling!.. Noon  Evening Night |
| CoreJava-Keywords- 018 | Consider the following Statements:  Statement A: If Junit test method is declared as Private the code will compile successfully but will throw Runtime exception  Statement B: In Junit all test methods must be declared to return "void"  Which of the following option is valid about the above statements? | Statement A and B are True |

|  |  |  |
| --- | --- | --- |
| CoreJava-Keywords- 019 | Consider the following Code:  class TestWorld {  public static void main(String args[]) { int N;  N = 1;  while ( (N <<= 32) >= 0) { N = 2 \* N;  }  System.out.println(N);  }  }  Which of the following option gives the output for the above code? | Integer.MIN\_VALUE |
| CoreJava-Keywords- 020 | Consider the following code:   1. public class GuessArraySize { 2. public static void main (String args[]) { 3 int array[] = {1, 2, 5}; 3. System.out.println("FirstGuess: " + 4. ((int[])fillArray(array)).length); 5. System.out.println("SecondGuess: " + 6. ((String[])fillArray(args)).length); 8 } 7. private static Object fillArray(Object original) { 8. Object returnValue = null; 9. return returnValue; 12 }   13 }  Which of the following option gives the output of the above code when following command line arguments are passed for the program "java" "and" "java" "always"? | Runtime error at line no 5 |

|  |  |  |
| --- | --- | --- |
| CoreJava-Keywords- 021 | Consider the following Code:  public class question2 {  public static void main(String args[]) { Float f1 = new Float("1.4e99f"); Float f2 = new Float("-2.4e99f"); Double d1 = new Double("3.4e99"); System.out.println(f1); System.out.println(f2); System.out.println(d1);  }  }  Which of the following option gives the output for the above code? | Infinity  -Infinity 3.4E99 |
| CoreJava-Keywords- 022 | Consider the following Code:  class Testone { public class Main {  public static void main(String[] args) { int i=1;  int n=++i%5; System.out.println("value of n is:"+n); n=i--%4;  System.out.println("value of n is:"+n); n=i++%2;  System.out.println("value of n is:"+n);  }  }  Which of the following option gives the output for the above code? | 2,2,1 |

|  |  |  |
| --- | --- | --- |
| CoreJava-Keywords- 023 | Consider the following code:  public class QuestionOpeAssign { static int num=5;  public static void main(String[] args) { int sum= (num=3) \* num;  num=6; System.out.println(sum); System.out.println(num);  }  )  Which of the following option gives the output for the above code? | 9,6 |
| CoreJava-Keywords- 024 | Consider the following code:  public class sortInt {  public static void main(String args[]) { short a=7;  byte v=6; int k=a+v;  System.out.println("value of "+k);  }  }  Consider the following statements regarding the above code:  A)The above code will throw a compile time error at line no 6  B)The above code will execute successfully if line no 6 is replaced by short k=(short)a+v;  C)The above code will execute successfully if line no 6 is replaced by byte k=(byte)a+v;  D)The above code will execute successfully if line no 6 is replaced by int k=a+v;  Which of the following option is valid regarding the above code and statements? | Only A is correct |

|  |  |  |
| --- | --- | --- |
| CoreJava-Keywords- 025 | Consider the following code:  public class MethodConversion {  static int m(byte a, int b) { return a+b; } static int m(short a, short b) { return a-b; } public static void main(String[] args) {  System.out.println("Value Returned form the method is:"+m(12, 2));  }  }  Which of the following option gives the output for the above code? | Compile time error at line number 5 |
| CoreJava-Keywords- 026 | Consider the following code:  enum Vehicle {  Bike ("Two wheeler"), Car ("Four Wheeler"); String type;  Vehicle (String type) { this.type = type;  }}  public class TestEnum {  public static void main (String[] args) { Vehicle car = new Vehicle ("Four Wheeler"); System.out.println ( car );  }}  Which of the following option gives the output for the above code? | Throws a Compiletime error as Enums may not be instantiated. |

|  |  |  |
| --- | --- | --- |
| CoreJava-Keywords- 027 | Consider the following code:  enum Games{ Cricket, FootBall, Hockey;  abstract String rules();  }  class Play{  public static void main (String[] args) { System.out.println (FootBall.rules);  }}  Which of the following statement is true regarding the above code? | The above mentioned code will fail to compile. |
| CoreJava-Keywords- 028 | Consider the following code:  public class Array4 {  public static void main(String[] args) throws Throwable  {  int me[][] = { { 1 , 2}, null };  int you[][] = me.clone(); System.out.print((me == you));  System.out.println(me[0] == you[0] && me[1] == you[1]);  }  }  Statement A:Line 5 prints true Statement B:line 6 Prints true  Which of the following option is valid regarding the above statements and code? | A is false and B is true |
| CoreJava-Keywords- 029 | Consider the following Statement: int[] values = new int[10] ;  Which of the following option is true about the above declaration? | It declares values to be a reference to an array object and constructs an array object containing 10 integers which are initialized to zero. |

|  |  |  |
| --- | --- | --- |
| CoreJava-Keywords- 030 | Consider the following code:  Line No 1:public class DoubleDemoArray {  Line No 2: public static void main (String args[]) { Line No 3: int array1[] = {1, 2, 3, 4, 5};  Line No 4: int array2[] = {1, 2, 3, 4, 5, 6, 7, 8, 9}; Line No 5: System.out.println("New Array1 size: " + doubleArray(array1).length);  Line No 6: System.out.println("New Array2 size: " + doubleArray(array2).length);  Line No 7: }  Line No 8: private static int[] doubleArray(int original[]) { Line No 9: int length = original.length;  Line No 10: int newArray[] = new int[length\*2];  Line No 11: System.arraycopy(original, 4, newArray, 2, length-4);  Line No 12: return newArray; Line No 13: }  Line No 14:}  Which of the following option gives the output for the above code? | 10,18 |
| CoreJava-Keywords- 031 | Consider the following code snippet:  Line No:1 public class Array3 {  Line No:2 public static void main(String args[]){ Line No:3 int ia[][] = { {1, 2}, null };  Line No:4 for (int[] ea : ia) Line No:5 for (int e: ea)  Line No:6 System.out.print(e); Line No:7 }}  Which of the following option gives the output for the above code? | Runtime error at line number 5 |
| CoreJava-Keywords- 032 | Consider the following declaration of a String Array with name as noName:  String[] noName = new String[10] ;  Which of the following option gives the partial code fragments that prints out the slots of the array from last to first, skipping slots that contain null? | for ( int j = names.length-1; j >= 0; j-- )  if ( names[j] != null ) System.out.println(  names[j] ); |

|  |  |  |
| --- | --- | --- |
| CoreJava-Keywords- 033 | Consider the following code:  import java.util.\*; enum Grade {  YELLOW (Personality.EXPRESSIVE), GREEN (Personality.AMIABLE), RED (Personality.ASSERTIVE), BLUE (Personality.ANALYTICAL); Personality personality;  Grade (Personality personality) { this.personality = personality;  }  enum Personality {ASSERTIVE, EXPRESSIVE, AMIABLE, ANALYTICAL };  }  public class FindGrade{  public static void main (String[] args) {  // INSERT LINE OF CODE HERE  }  }  Which of the following line of code when replaced with "// INSERT LINE OF CODE HERE", will give the output value "true"? | System.out.println (Grade.Personality.ASSERTI VE instanceof Grade.Personality); |

|  |  |  |
| --- | --- | --- |
| CoreJava-Keywords- 034 | Consider the following code:  enum PerformanceRating { POOR (0.0, 5.0),  AVERAGE (5.1, 7.0),  GOOD (7.0, 8.5),  EXCELLENT (8.6, 9.9);  double Min, Max;  PerformanceRating (double Min, double Max) { this.Min = Min;  this.Max = Max;  }  int awardMarks(){ switch (this) {  case POOR: return 0; case AVERAGE: return 5; case GOOD: return 20;  case EXCELLENT: return 45;  }  return 0;  }  }  public class Appraisal { | POOR Performance:Revised Salary = 100.0  AVERAGE  Performance:Revised Salary = 105.0  GOOD  Performance:Revised Salary = 120.0 EXCELLENT  Performance:Revised Salary = 145.0 |

|  |  |  |
| --- | --- | --- |
| CoreJava-Keywords- 035 | Consider the following code:  enum Currency {  Ruppee(1), Doller(5), Dinar(10); Currency(int value) { this.value = value; } private final int value;  public int value() { return value; }  }  public class EnumDifficult {  public static void main(String[] args) { Currency[] c=Currency.values();  for(int i=0;i<Currency.values().length;i++)  {  if(i==2) break;  System.out.println( c[i] +"\t" + color(c[i+1]));  }}  private enum CurrencyColor { Blue, Red, Yellow } private static CurrencyColor color(Currency c) {  if (c == null)  throw new NullPointerException();  switch(c) { case Ruppee: | Ruppee Red  Doller Yellow |

|  |  |  |
| --- | --- | --- |
| CoreJava-Keywords- 036 | Consider the following code snippet:  public class checkValues {  Line No 1:static int numOne=20; Line No 2:int numTwo=10;  Line No 3:public static void main(String argv[]){ Line No 4: int numTwo=5;  Line No 5: checkValues p = new checkValues(); Line No 6: p.changeValue(numTwo);  Line No 7: System.out.println(numTwo); Line No 8: System.out.println(numOne); Line No 9: }  Line No 10: public void changeValue(int numOne){ Line No 11: numTwo=numOne\*2;  Line No 12: numOne=numOne\*2; Line No 13: }  Line No 14:}  Which of the following option gives the output at Line numbers 7 and 8 ? | 5,20 |
| CoreJava-Keywords- 037 | Consider the following code:  Line No 1:class GardenFlower {  Line No 2:public static void main(String [] args) { Line No 3:int numOfFlowers=20;  Line No 4:int sizeOfGarden = 10;  Line No 5:sizeOfGarden=sizeOfGarden<<8;  Line No 6:String status = (numOfFlowers<4)?"Flowers count OK"  ::(sizeOfGarden < 8)? "Garden limit on the edge"  :"too many Flowers";  Line No 7:System.out.println("Garden status is " + status);  Line No 8:}  Line No 9:}  Which of the following option gives the output for above code? | Garden status is too many Flowers |

|  |  |  |
| --- | --- | --- |
| CoreJava-Keywords- 038 | Consider the following code:  public class ConvertionCastAvg1 { public static void main(String[] args) {  float fmin = Float.NEGATIVE\_INFINITY; float fmax = Float.POSITIVE\_INFINITY;  System.out.println("short: " + (short)fmin + ".." + (short)fmax);  System.out.println("byte: " + (byte)fmin +".." + (byte)fmax);  }  }  Which of the following option gives the output for the above code? | short:0..1 byte:0..1 |
| CoreJava-Keywords- 039 | Consider the following code snippet: class PointValue { int x, y; }  interface Flavour { void setFlavour(int flavour); }  class FlavouredPoint extends PointValue implements Flavour  {  int flavour;  public void setFlavour(int flavor) { this.flavour = flavour; }  }  final class EndFlavourPoint extends PointValue { } Line no:1 class CastingTest {  Line no:2 public static void main(String[] args) { Line no:3 PointValue p = new PointValue();  Line no:4 FlavouredPoint cp = new FlavouredPoint();  Line no:5 Flavour c;  Line no:6 cp = (FlavouredPoint)p;  Line no:7 c = (Flavour)p;  Line no:8 EndFlavourPoint e = new EndFlavourPoint();  Line no:9 c=(Flavour)e; | Code generates compile time error at line no 7 |

|  |  |  |
| --- | --- | --- |
| CoreJava-Strings-001 | Which of the following options are the methods NOT available in StringBuffer class?   1. append(boolean b) 2. append(byte b) 3. append(short s) 4. append(int i) 5. append(long l) | 2,3 |
| CoreJava-Strings-002 | Consider the following code snippet:  StringBuffer sbr = new StringBuffer(); System.out.print(sbr.capacity());  sbr.append("Think Green").append("Think Green"); System.out.println(sbr.capacity());  Which of the following option gives the output of the above code snippet? | 1634 |
| CoreJava-Strings-003 | Which of the following statement gives the significance of overriding equals() method in user defined classes? | Comparing memory references of two objects |
| CoreJava-Strings-004 | Which of the following statement gives the significance of overriding hashCode() method in user defined classes? | Ensuring uniqueness of the objects being created |
| CoreJava-Strings-005 | Consider the following code snippet:   1. String thirdBinded = "BINDED"; 2. String bindedString = new String("Binded"); 3. String secondBinded = bindedString.toUpperCase();   Which of the following option gives correct lines number of the statements in the above code, that uses the JVM's String Object Pool? | Line 1 |
| CoreJava-Strings-006 | Which of the following are NOT a default delimiter of StringTokenizer class?   1. , (comma) 2. \t (tab) 3. \n (new line) 4. ; (semi colon) 5. \f (form feed) | 1,4 |

|  |  |  |
| --- | --- | --- |
| CoreJava-Strings-007 | Which of the following option gives the name of the Exception which is thrown when a String with Non- Numeric value is parsed with Integer.valueOf() method? | NumberFormatException |
| CoreJava-Strings-008 | Which of the following options give the member methods of String class that creates new String object?   1. toString() 2. concat() 3. startsWith() 4. trim() 5. endsWith() | 2,4 |
| CoreJava-Strings-009 | Which of the following options give the methods that are not member of String class?   1. length() 2. capacity() 3. trim() 4. delete() 5. replace() | 2,4 |
| CoreJava-Strings-010 | Which of the following option gives the member method of String class that does not create a new String object? | toString() |
| CoreJava-Strings-011 | Which of the following option is the member method of String class, that adds the String object to the JVM's Object Pool? | intern() |
| CoreJava-Strings-012 | Consider the following code snippets:  Code Snippet 1:  String part1 = new String("Thought"); String part2 = new String("Green"); String part3 = new String("World"); String part4 = new String("Green");  String fullString = part1 + " is " + part2 + ", so the " + part3 + " is " + part4;  Code Snippet 2:  String part1 = "Thought"; String part2 = "Green"; String part3 = "World"; String part4 = "Green";  String fullString = part1 + " is " + part2 + ", so the " + part3 + " is " + part4;  Which of the following statement is true regarding the above code snippets? | Code Snippet 1 uses more JVM memory than Code Snippet 2 for storage |

|  |  |  |
| --- | --- | --- |
| CoreJava-Strings-013 | Which of the following statements are TRUE about StringBuffer class?   1. StringBuffer is a mutable class 2. StringBuffer can be extended, since it is mutable 3. StringBuffer is a sub class of String class 4. StringBuffer is a Wrapper to the existing String class 5. StringBuffer class can instantiate String type objects | 1,5 |
| CoreJava-Strings-014 | Consider the following code snippet:  StringBuffer thought = new StringBuffer("Green"); thought  .insert(0, "Ever")  .insert(0, "PlanetEarthIs")  .delete(0, 14); System.out.println(thought.capacity());  Which of the following option gives the output of the above code snippet? | 44 |
| CoreJava-Strings-015 | Consider the following code snippet:   1. String truth = "null"; 2. Boolean truthValue = Boolean.valueOf(truth); 3. System.out.println(truthValue);   Which of the following option gives the output of the above code snippet? | Prints false |
| CoreJava-Strings-016 | Consider the following code snippet:   1. String truth = null; 2. String truthValue = String.valueOf(truth); 3. System.out.println(truthValue.equals("null"));   Which of the following option gives the output of the above code snippet? | Prints true |
| CoreJava-Strings-017 | Which of the following option gives the name of the property that builds the equivalence relation of equals() method of an Object, that directly depends on the output of its hashCode() method? | consistency |

|  |  |  |
| --- | --- | --- |
| CoreJava-Strings-018 | Which of the following statement gives the exact relationship between the equals() method and hashCode() method in the Object class? | There is no connection between equals() method and hashCode() method in the Object class. The equals() simply compares only the memory references of current object and object to be compared using == operator. |
| CoreJava-Strings-019 | Which of the following statement is TRUE regarding overriding equals() and hashCode() methods? | Both the methods equals() and hashCode() has to be overridden at the same time |
| CoreJava-Strings-020 | Consider the following code snippet:  StringBuffer game = new StringBuffer(""); game  .insert(0, "Play")  .insert(0, "With")  .insert(0, "Pea")  .insert(0, "Nuts"); game.delete(0, 4); System.out.println(game);  Which of the following option gives the output of the above code snippet? | PeaWithPlay |

|  |  |  |
| --- | --- | --- |
| CoreJava-Strings-021 | Consider the following code snippet:  StringBuffer game = new StringBuffer(""); game  .append("Play")  .append("With")  .append("Pea")  .append("Nuts"); game.delete(0, 4); System.out.println(game);  Which of the following option gives the output of the above code snippet? | WithPeaNuts |
| CoreJava-Strings-022 | Consider the following code snippet:  String greenThought =  "The thought is Green, so is the World"; if(greenThought.substring(  greenThought.indexOf("th"), greenThought.lastIndexOf("is")  ).startsWith("o")) { System.out.println(greenThought.substring(  greenThought.indexOf("Green")  ));  } else { System.out.println(greenThought.substring(  0, greenThought.indexOf("Green")  ));  }  Which of the following option gives the correct output of the above code snippet? | The Thought is |

|  |  |  |
| --- | --- | --- |
| CoreJava-Strings-023 | Consider the following code snippet:  String greenWorld =  "The World is Green, so is the Thought"; if(greenWorld.substring(  greenWorld.indexOf("Th"), greenWorld.lastIndexOf("is")  ).startsWith("o")) { System.out.println(greenWorld.substring(  greenWorld.indexOf("Thought")  ));  } else { System.out.println(greenWorld.substring(  0, greenWorld.indexOf("Thought")  ));  }  Which of the following option gives the number of String objects, that gets created in the above code snippet? | 8 |
| CoreJava-Strings-024 | Consider the following code snippet:   1. String thought = "A Lion or\t\ta deer\n, better\f \fbe running"; 2. StringTokenizer tokenizer = new StringTokenizer(thought); 3. System.out.println(tokenizer.countTokens());   Which of the following option gives the output of the above code snippet? | 9 |
| CoreJava-Strings-025 | Consider the following code snippet:   1. int count = 0; 2. String thought = "This\t\\tis intentionally\n\n made tough"; 3. StringTokenizer tokenizer = new StringTokenizer(thought);   4  for(count=0;tokenizer.hasMoreTokens();tokenizer.nextT oken("t"), count++);  5 System.out.println(count);  Which of the following option gives the output of the above code snippet? | 5 |

|  |  |  |
| --- | --- | --- |
| CoreJava-Strings-026 | Consider the following code:  public class MyClass { private int a; private int b;  MyClass() { } MyClass(int a, int b) {  this.a = a; this.b = b;  }  }  Which of the following option gives the correct way of implementing the equals() method for the above class? | public boolean equals(Object obj) {  if(obj != null && obj instanceof MyClass) {  MyClass that = (MyClass) obj;  return (this.a == that.a) && (this.b == that.b);  }  return false;  } |
| CoreJava-Strings-027 | Consider the following code:  public class Employee { private Integer empId; private String empName;  public Employee() { }  public Employee(Integer empId, String empName) { this.empId = empId;  this.empName = empName;  }  }  Which of the following option gives the correct implementation of hashCode() method, that helps a HashSet to add only the unique employee objects? | public int hashCode() { return  (this.empId.hashCode() \* 2)  +  (this.empName.hashCode()  \* 3);  } |

|  |  |  |
| --- | --- | --- |
| CoreJava-Strings-028 | Consider the following code:  public class World { private String name;  public World() { }  public World(String name) { this.name = name;  }  }  Which of the following option gives the correct implementation of hashCode() method, that does not violate the Equals and Hash Code relationship? | public int hashCode() { return  name.hashCode();  } |
| CoreJava-Strings-029 | Consider the following code:  public class GeoCode { private Double longitude; private Double latitude;  public GeoCode() { }  public GeoCode(Double longitude, Double latitude) { this.longitude = longitude;  this.latitude = latitude;  }  }  Which of the following implementation of hashCode() methods satisfies the basic requirement of the equals() method in relation with hashCode() method? | public int hashCode() { return  longitude.hashCode() + latitude.hashCode();  } |
| CoreJava-Threads-001 | Consider the following statements about a demon Thread :  A demon Thread group   1. has only demon threads. 2. can have non demon threads. 3. does not exist after all non demon threads in the group have finished executing. 4. does not exist after all the threads in the group have finished executing.   Which of the following option gives the statements that are true for above? | 2, 4 |

|  |  |  |
| --- | --- | --- |
| CoreJava-Threads-002 | Consider the following scenario:  In a File System application, there are two threads - one called readthread which will read a file and the other called writethread which will write to a file. The readthread waits for the writethread for its execution.  arises when the writethtread also waits at the same time.  Fill in the blank with the valid answer from the below given options | DeadLock Condition |
| CoreJava-Threads-003 | Which of the following option gives the name of the method that informs the current thread to leave the control of the monitor? | wait() |
| CoreJava-Threads-004 | Consider the following Statements:  Statement A:  wait, notify and notifyAll methods are not called on Thread, they are called on Object  Statement B:  These methods can only be called from synchronized code, or an IllegalMonitorStateException will be thrown.  Which of the following option is true regarding the above statements? | Statement A and B both are true |
| CoreJava-Threads-005 | Which of the following statements are true regarding threads and classes?   1. The class Thread is abstract. 2. The class Thread implements Runnable. 3. Classes implementing the Runnable interface must define a method named start. 4. Calling the method run() on an object implementing Runnable will create a new thread. 5. A program terminates when the last non-daemon thread ends. | 2,5 |
| CoreJava-Threads-006 | Consider the following scenario:  Two Threads A and B hit two different synchronized methods in an object at the same time. Will they both continue?  Which of the following option gives the valid outcome for the above question? | No, only one method can acquire the lock |

|  |  |  |
| --- | --- | --- |
| CoreJava-Threads-007 | Consider the following code:  class Apple implements Runnable{public void run() {}} class Banana {  public static void main(String[] args) {  Thread t1 = new Thread(); // Line no 1  Thread t2 = new Thread(new Apple()); // Line no 2 Thread t3 = new Thread(new Apple(), "Apple"); // Line  no 3  Thread t4 = new Thread("Apple"); // Line no 4  }}  Which of the following option gives the line number at which a compile-time error is generated in the above program? | None of the listed options |
| CoreJava-Threads-008 | Consider the following statements:  Statement A: The priority of a thread can be set by using the setPriority() method in the Thread class.  Statement B: The priority of a thread can be set by passing the priority as a parameter to the constructor of the thread.  Which of the following option is true regarding the above statements? | Statement A and B both are true |

|  |  |  |
| --- | --- | --- |
| CoreJava-Threads-009 | Consider the following code:  class Fruit extends Thread  {  public void run()  {  // Line no 1 System.out.println(name);  }  }  public class ThreadName {  public static void main(String args[])  {  Fruit f =new Fruit(); f.setName("Apple"); f.start();  }})  Which of the following code snippets when replaced at Line no 1 will print the output as “Apple”?   1. String name=Thread.currentThread().getName(); 2. String name=this.currentThread().getName(); 3. String name=Thread.getInstance().currentThread().getName(); | 1,4 |

|  |  |  |
| --- | --- | --- |
| CoreJava-Threads-010 | Consider the following code:  public class RunningThread implements Runnable  {  public void run()  {  while(true)  {  System.out.println("I am RunningThread");  }  }  public static void main(String args[])  {  RunningThread nt1 = new RunningThread(); RunningThread nt2 = new RunningThread(); RunningThread nt3 = new RunningThread(); nt1.run();  nt2.run();  nt3.run();  }  }  Which of the following option gives the valid output for the above program? | Prints I am RunningThread infinitely. |

|  |  |  |
| --- | --- | --- |
| CoreJava-Threads-011 | Consider the following partial code:  public synchronized void work()  {  try{ Thread.sleep(2000);  }catch(InterruptedException e)  {  e.printStactTrace();  }  }  Which of the following options are true regarding the above code?   1. The code causes compilation error - sleep cannot be called inside synchronized methods. 2. The code causes compilation error - sleep is not a static method of java.lang.Thread 3. The Thread sleeps for at least 2000 milliseconds in this method if not interrupted. 4. When the thread "goes to sleep" it releases the lock on the object. 5. The "sleeping" Threads always have the lock on the Object. | 3,5 |
| CoreJava-Threads-012 | Consider the following code:  public class Play implements Runnable {  public void run() { System.out.print("Play");  }  public static void main(String[] args) { Thread t = new Thread(new Play()); t.start();   * 1. un();//Line no 1 t.run();   2. tart(); //Line no 2   }  }  Which of the following option gives the valid output for the above program? | prints PlayPlayPlay and throws a Runtime Exception |

|  |  |  |
| --- | --- | --- |
| CoreJava-Threads-013 | Consider the following code:  public class Swim implements Runnable{ public static void main(String argv[]){ Swim s = new Swim();  s.go();  }  public void go(){  //Line no 1{code to be replaced here}  }  public void run(){ System.out.println("swim");  }  }  Which of the following option gives a valid code snippet, when replaced at Line no 1 will generate the output as swim? | Thread t = new Thread(this);  t.start(); |
| CoreJava-Threads-014 | Consider the following code:  class PrintData extends Thread { private int i;  public void run() {i = 1;}  public static void main(String[] args) { PrintData a = new PrintData(); a.start();  try { a.join();  } catch (InterruptedException e) {  // TODO Auto-generated catch block e.printStackTrace();  }  System.out.print(a.i);//Line no 1  }}  Which of the following option gives the valid output for the above code? | Prints 0 |

|  |  |  |
| --- | --- | --- |
| CoreJava-Threads-015 | Consider the following code:  public class Thread6 {  public static void main(String [] args) { System.out.print("1 "); synchronized(args){ System.out.print("2 ");  try {  args.wait();//Line no 1  }  catch(InterruptedException e){ System.out.print("3");  }  }  System.out.print("4");  }  }  Which of the following option gives the valid output for the above code? | 12 |
| CoreJava-Threads-016 | Assume that in a class named Count there are two integer private data members:num1 and num2.  Which of the following options give the valid code that can prevent concurrent access in the Count class?   1. public int read(int a, int b){return a+b;}   public void Write(int a, int b){this.a=a;this.b=b;}   1. public synchronized int read(int a, int b){return a+b;} public synchronized void Write (int a, int   b){this.a=a;this.b=b;}   1. public int read(int a, int b){synchronized(a){return a+b;}}   public void Write (int a, int b){synchronized(a){this.a=a;this.b=b;}}   1. public int read(int a, int b){synchronized(a){return a+b;}}   public void Write (int a, int b){synchronized(b){this.a=a;this.b=b;}}   1. public int read(int a, int b){synchronized(this){return a+b;}}   public void Write (int a, int b){synchronized(this){this.a=a;this.b=b;}} | 2,5 |

|  |  |  |
| --- | --- | --- |
| CoreJava-Threads-017 | Consider the following statements:  Statement A: IllegalThreadStateException is a Checked Exception  Statement B: InterruptedException is UnChecked Exception  Which of the following option is true regarding the above given statements? | Both Statement A and B are False |
| CoreJava-Threads-018 | Consider the following scenario:  After invoking the wait method on an object obj1, a thread T1, will remain in the wait set of obj1.  Which of following are valid events that can occur for the above scenario to happen?   1. Another thread invokes the notify method on the object, obj1, and T1 is selected to move out of the wait set 2. Another thread invokes the join method on thread T1 3. A specified timeout period has elapsed 4. Another thread interrupts thread T1 5. The priority of thread T1 is increased | 1,3,4 |
| CoreJava-Threads-019 | Consider the following code:  public class Test extends Thread{ public static void main(String argv[]){ Test b = new Test();  b.run();  }  public void start(){  for (int i = 0; i <10; i++){ System.out.println("Value of i = " + i);  }  }  }  Which of the following option gives the valid output for the above given code? | Clean compile but no output at runtime |

|  |  |  |
| --- | --- | --- |
| CoreJava-Threads-020 | Consider the following code:  public class TestClass  {  public Charlie charlie = new Charlie(); public Tango tango = new Tango();  class Charlie  {  public synchronized void doIt()  {  try{  wait(); //LINE 1 System.out.println("done");}  catch(Exception e) { e.printStackTrace();}  }  }  class Tango extends Thread  {  public void run()  { charlie.doIt(); }  }  public TestClass() throws Exception  {  tango.start(); | synchronized(charlie){ charlie.notifyAll(); } |

|  |  |  |
| --- | --- | --- |
| CoreJava-Threads-021 | Consider the following code:  class Movie extends Thread { Movie() { System.out.print("Nature");  }  public void run() { try {  Thread.sleep(1000); run("StarWars");  } catch (InterruptedException e) {  // TODO Auto-generated catch block e.printStackTrace();  }  System.out.print("Motivates");}  public void run (String s) { System.out.print(s); }  }  public class Director {  public static void main (String [] args) { Thread t = new Movie(){  public void run() { run("Titanic");  System.out.print("GivesEnergy"); }  //Line No 1: }; t.start(); | NatureTitanicGivesEnergy |

|  |  |  |
| --- | --- | --- |
| CoreJava-Threads-022 | Consider the following code:  public class Print extends Thread { Thread t;  Print() { } Print(Thread t) {  this.t = t;  }  public static void main(String[] args) {  new Thread(new Print(new Print())).start();  }  public void run() {  System.out.println("Honey"); new Thread(t).start(); System.out.println("Bee");  }  }  Which of the following option give the valid output for the above code? | HoneyBeeHoneyBee |

|  |  |  |
| --- | --- | --- |
| CoreJava-Threads-023 | Consider the following code:  class Good extends Thread { public void run()  {  System.out.print("Good");  }  }  class Best extends Thread  {  public void run()throws AnException{ //Line no 1  {  System.out.print("Best");  throws new AnException();  }  }  public class TestThread {  public static void main(String[] args) { Good a = new Good();  a.start();  Best b=new Best(); b.start(); //Line no 2  } | Compilation Error at Line no 1 |

|  |  |  |
| --- | --- | --- |
| CoreJava-Threads-024 | Consider the following code :  public class Car extends Thread { static String cName = "Thunder";  public static void main(String argv[]){ Car t = new Car(); t.getName(cName);  }  public void getName(String cName){ cName = cName + "Bolt";  start();//Line no 1 System.out.println(cName);  }  public void run(){  for(int i=0;i < 4; i++){ cName = cName + "V" + i;  }  }}  Which of the following option gives the valid output for the above code? | Compilation and output of either " Thunder Bolt ", " Thunder Bolt 0", " Thunder Bolt 0 1" " Thunder Bolt 0 1  2" or " Thunder Bolt 0 1 2  3" |

|  |  |  |
| --- | --- | --- |
| CoreJava-Threads-025 | Consider the following code:  class Tree implements Runnable{ public void run()  {  System.out.println("RunningTree");  }  public void start()  {  System.out.println("StartingTree");  }  }  public class Plantation extends Thread  {  public static void main(String argv[]){ Runnable r=new Tree();  Thread t=new Thread(r); t.start();//Line no 1  }  public void run(){  for (int i = 0; i <3; i++){ System.out.println("GreenPlanet");  }  }  } | RunningTree |

|  |  |  |
| --- | --- | --- |
| CoreJava-Threads-026 | Consider the following code: class Machine  {  public void run()  {  System.out.println("MachineRunning");  }  }  class Factory extends Thread { public Machine man;  public void run() { try {  synchronized(man) { System.out.println("Guns"); man.wait(); Thread.sleep(1000); man.notify(); System.out.println("Pistols");  }  } catch(InterruptedException e) { }  }  }  public class TestWait {  public static void main(String[] args) { Machine m = new Machine(); | Prints Planet and throws a Runtime Exception |

|  |  |  |
| --- | --- | --- |
| CoreJava-Threads-027 | Consider the following code:  public class TestThreads implements Runnable { volatile int i=0;  public void run()  {  while(true)  {i++; //Line no 1  }  System.out.println(i);  }  public static void main(String args[]) { TestThreads t1 = new TestThreads(); TestThreads t2 = new TestThreads(); t1.run();  t2.run();  }}  Which of the following option gives the valid output for the above code? | Compiles and runs without any output |
| CoreJava-Threads-028 | Which of the following statements are true with respect to threads?   1. Once a new thread is started, it will always enter the runnable state 2. You can call start() on a Thread object only once. If start() is called more than once on a Thread object, it will throw a CompileTime Excpetion. 3. The sleep() method is a static method that sleeps the currently executing thread. One thread cannot tell another thread to sleep. 4. The setPriority() method is used on Thread objects to give threads a priority of between 1 (high) and 10 (low) 5. The notify() method is used to send a signal to all the threads that are waiting in that same object’s waiting pool. | 1,3 |

|  |  |  |
| --- | --- | --- |
| CoreJava-Threads-029 | Consider the following code:  class WaitTime extends Thread { public void run() { synchronized (this) {  try {wait(5000);} catch (InterruptedException ie){}  }}  public static void main(String[] args) { WaitTime wt = new WaitTime();  long startTime = System.currentTimeMillis(); wt.start(); System.out.print(System.currentTimeMillis() -  startTime + ",");  try {wt.join(6000);} catch (InterruptedException ie) {} System.out.print(System.currentTimeMillis() -  startTime);  }}  Which of the following options are true regarding the above code?   1. The first number printed is greater than or equal to 0 2. The first number printed must always be greater than 5000 3. The second number printed must always be greater than or equal to 5000 4. The second number printed must always be greater | 1,3 |

|  |  |  |
| --- | --- | --- |
| CoreJava-Threads-030 | Consider the following code:  class Print extends Thread { String[] data;  public Print(String[] data) {this.data = data;} public void run() {  synchronized (data) {System.out.print(data[0] + data[1]  + data[2]);}  }}  public class Test {  private static String[] data = new String[]{"X","Y","Z"}; public static void main (String[] args) {  synchronized (data) {  Thread t1 = new Print(data); t1.start(); data[0] = "A"; data[1] = "B"; data[2] = "C";  }}}  Which of the following option gives the valid output for the above code? | Prints: ABC |
| CoreJava-Threads-031 | Consider the following code:  public class A extends Thread { public void run() {  yield();//Line no 1 System.out.println("world");  }  public static void main(String[] args) { A a1 = new A();  a1.start(); try  {  sleep(1000);  }catch(Exception e)  {  e.printStackTrace();  }  a1.join();//Line no 2 System.out.println("hello");  }}  Which of the following options gives the valid output for the above code? | Compilation Error at Line no 2 |

|  |  |  |
| --- | --- | --- |
| CoreJava-Threads-032 | Consider the following code:  class A extends Thread { private static B b = new B(); private String s1;  public void run() {System.out.print(b.m1(s1));} A(String threadName, String s1) { super(threadName); this.s1 = s1;  }  public static void main (String[] args) {  A a = new A("T1","A"), b = new A("T2","B"); a.start(); b.start();  }}  class B {  private String s1;  public synchronized String m1(String s) { s1 = s;  try {Thread.sleep(1);} catch (InterruptedException ie) {} return  "["+Thread.currentThread().getName()+","+s1+"]";  }}  Which of the following options give valid outputs for the above code?   1. Prints: [T1,A][T2,B] 2. Prints: [T1,B][T2,B] | 1,3 |

|  |  |  |
| --- | --- | --- |
| CoreJava-Threads-033 | Consider the following code:  class FirstThread extends Thread { public static void main(String [] args) { FirstThread t = new FirstThread(); t.setName("First");  Thread a= new Thread(t,"Second"); a.start();  }  public void run() { for(int i=0;i<3;++i) {  System.out.println("CAT");  }  this.start();  }  }  Which of the following option gives the valid output for the above code? | Prints CAT Six times and throws a Runtime Exception |
| CoreJava-Threads-034 | The synchronized statement has the form shown below. synchronized (expression) block  Which of the following are true statements with respect to synchronization?  Statement A:  If execution of the block completes normally, then the lock is released  Statement B:  If execution of the block completes abruptly, then the lock is released  Which of the following option is true regarding the above given statements? | Both Statement A and B are True |

|  |  |  |
| --- | --- | --- |
| CoreJava-Threads-035 | Which of the following statements are true regarding threads?   1. A program will terminate only when all daemon threads stop running 2. A program will terminate only when all user threads stop running 3. A daemon thread always runs at Thread.MIN\_PRIORITY 4. A thread inherits its daemon status from the thread that created it 5. The daemon status of a thread can be changed at any time using the Thread.setDaemon method | 2,4 |
| CoreJava-Threads-036 | Consider the following code:  interface MyRunnable extends Runnable { } interface MyRunnable2 {  public abstract void run();  }  class MyThread implements MyRunnable2 { public void run() {  try {  System.out.println("Thread Started"); Thread.sleep(3000); System.out.println("Thread Ended");  } catch(InterruptedException ie) { System.out.println("Thread interrupted:" +  ie.getMessage());  }  }  }  public class TestRunnable {  public static void main(String[] args) {  Thread t = new Thread(new MyThread()); /\* ERROR HERE \*/  t.start();  System.out.println("Program Ended"); | 1,3 |

|  |  |  |
| --- | --- | --- |
| CoreJava-Threads-037 | Consider the following code:  class Search implements Runnable { Search search;  public void search() { System.out.println("Searching...");  new Thread(this.search).start(); /\* LINE 1 \*/  }  public void run() { try {  Thread.currentThread().sleep(3000); this.search(); /\* LINE 2 \*/  } catch(InterruptedException ie) { System.out.println("Thread interrupted");  }  }  }  public class TestSearch {  public static void main(String[] args) { Search search = new Search(); search.search = search;  new Thread(search).start(); /\* LINE 3 \*/  }  } | 1,2 |

|  |  |  |
| --- | --- | --- |
| CoreJava-Threads-038 | Consider the following code:  class Task {  public void run() { System.out.println("Task started"); try {  Thread.currentThread().sleep(2000);  }catch(InterruptedException ie) { System.out.println("Thread interrupted");  }  System.out.println("Task completed");  }  }  class RunnableTask extends Task implements Runnable {  }  public class TestTask {  public static void main(String[] args) {  /\* CODE \*/  }  }  Which of the following code snippet when substituted at the line commented as /\* CODE \*/ in the above program will invoke the run() method? | new Thread(new RunnableTask()).start(); |

|  |  |  |
| --- | --- | --- |
| CoreJava-Threads-039 | Consider the following code:  class AB extends Thread {  private static String[] sa = new String[]{"Not Done","X","Y","Z"};  public void run() { synchronized (sa) {  while (!sa[0].equals("Done")) {  try {sa.wait();} catch (InterruptedException ie) {}  }  }  System.out.print(sa[1] + sa[2] + sa[3]);  }  public static void main (String[] args) { try {  Thread.sleep(1000);//Line no 1 AB a=new AB();  a.start();  } catch (InterruptedException e) {  // TODO Auto-generated catch block e.printStackTrace();  }  synchronized (sa) { sa[0] = "Done";  sa[1] = "A"; sa[2] = "B"; sa[3] = "C";  sa.notify();  }}} | ABC |
| CoreJava-Updations in SDK-001 | Which of the following class which is newly added in JDK 1.6, provides method to read password? | Console |
| CoreJava-Updations in SDK-002 | Which of the following method is newly added to the PrintStream class in JDK 1.6? | clearError() |
| CoreJava-Updations in SDK-003 | Which of the following option gives the code name for Java SE 6? | Mustang |
| CoreJava-Updations in SDK-004 | Which of the following option gives the name of the API which is newly added to Java SE 6? | Java Compiler API |
| CoreJava-Updations in SDK-005 | Which of the following option gives the newly added feature to Java SE 6? | JDBC 4.0 |
| CoreJava-Updations in SDK-006 | Which of the following option gives the name of the new type of parser added to the XML API, in JDK 1.6? | StAX |

|  |  |  |
| --- | --- | --- |
| CoreJava-Updations in SDK-007 | Which of the following option gives the name of the Collection implementation class that implements the newly introduced NavigableSet interface in JDK 1.6? | TreeSet |
| CoreJava-Updations in SDK-008 | Which of the following option gives the name of the package which is newly added to Java SE 6? | java.util.spi |
| CoreJava-Updations in SDK-009 | Which of the following option gives the name of the concrete implementation class, which is newly added to the collection framework in JDK 1.6? | ArrayDeque |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Wrong answer** | **Wrong answer** | **Wrong answer** | **Wrong answer** | **Marks** |
| A-III, B-I, C-II | A-I, B-II, C-III | A-II, B-I, C-III |  | 2 |
| Default Constructors are Optional for all classes | Can be overloaded across inherited classes | Abstract classes cannot have constructors |  | 2 |
| Parameterized constructors cannot accept its same class type as parameter | Parameterized constructors cannot accept final arguments as parameters | Parameterized constructors cannot call the default constructor |  | 2 |
| Compiles successfully without any error | Default constructors cannot be called from parameterized constructors | The this can be used only for accessing the member data and member methods, not constructors |  | 2 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A-III, B-II, C-I | A-I, B-II, C-III | A-II, B-I, C-III |  | 2 |
| Using a recursive method that accepts Var-args | Using method overriding | Using non-static overloaded methods |  | 2 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2,3 | 3,4 | 4,5 | 1,5 | 2 |
| 2,3 | 3,4 | 4,5 | 2,5 | 2 |
| A-II, B-I, C-III | A-I, B-III, C-II | A-I, B-II, C-III |  | 2 |
| Static methods cannot be overloaded | Overloaded methods should always return same type of value | Overloaded methods cannot be declared as abstract |  | 2 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1,2 | 2,3 | 3,4 | 1,3 | 2  2  2 |
| Both the length() methods are overloaded methods | Overloaded methods cannot start with a special character like '\_' | Overloaded methods should be declared as public |  |
| A-I, B-II, C-III | A-II, B-III, C-I | A-II, B-I, C-III |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2,3,4 | 3,4,5 | 1,2,4 | 2,4,5 | 2 |
| Private level access is applicable for both classes and its members | Public is applicable for local variables | Package level access is only for members, not for classes |  | 2 |
| 1,2 | 2,3 | 4,5 | 1,5 | 2 |

3

|  |  |  |  |
| --- | --- | --- | --- |
| Normal Version HiHelloWelcome Bye  Normal Version HiHelloWelcome ByeGood Night | Var-arg Version HiHelloWelcomeB ye  Var-arg Version HiHelloWelcomeB yeGood Night | Compile time error 'Ambiguity in resolving the method stringProcessor()' |  |

3

|  |  |  |
| --- | --- | --- |
| Array Version 100  Array Version 150 | Normal Version 100  Array Version 150 | Normal Version 100  Normal Version 100 |

3

|  |  |  |  |
| --- | --- | --- | --- |
| Array Version 10.0  Array Version 15.0 | Normal Version 10.0  Array Version 15.0 | Shows a compile time error for both the methods 'Duplicate methods numericProcessor(  ...)' |  |

3

|  |  |  |  |
| --- | --- | --- | --- |
| Built-out method: Hello Built-in method: Hello | Runtime Error 'Unable to resolve the method method(String)' | Built-out method: Hello  Built-in method: null |  |

3

|  |  |  |  |
| --- | --- | --- | --- |
| CODE 1 - Integer CODE 2 - Integer | CODE 1 - Float CODE 2 - Float | CODE 1 - String CODE 2 - String | CODE 1 - Number CODE 2 - Number |

3

|  |  |  |  |
| --- | --- | --- | --- |
| Object o[] | Integer id, String name, Double salary | Object id, Object name, Object salary |  |

3

|  |  |  |  |
| --- | --- | --- | --- |
| Student(s.id, s.name, s.salary); | new Student(s.id, s.name, s.salary); | Student(s); | Copy constructor is not possible in Java |

3

|  |  |  |  |
| --- | --- | --- | --- |
| Prints: Id:100  Name:Mr.X Salary:28000.0 | this() method call should be the first statement in a normal method | Cannot call overloaded constructors using this() method |  |

3

|  |  |  |  |
| --- | --- | --- | --- |
| 1,2 | 2,3 | 3,4 | 1,5 |

3

|  |  |  |
| --- | --- | --- |
| Prints Id:1  Name:Mr.Employ ee Salary:25000.0 | Runtime error | Compiles and executes successfully, but No Output |

3

|  |  |  |  |
| --- | --- | --- | --- |
| 1,2,3 | 2,3,4 | 3,4,5 | 1,3,4 |

3

|  |  |  |  |
| --- | --- | --- | --- |
| 2,3 | 3,4 | 4,5 | 1,5 |

3

|  |  |  |  |
| --- | --- | --- | --- |
| 2,3,4 | 1,2,5 | 3,4,5 | 2,4,5 |

3

|  |  |  |  |
| --- | --- | --- | --- |
| 10 | 11 | Class Not Found 11 |  |

3

|  |  |  |  |
| --- | --- | --- | --- |
| Class Not Found 100 | Class Not Found Class Not Found 100 | 102 |  |

3

|  |  |  |  |
| --- | --- | --- | --- |
| Class Not Found 1001 | 1001 | Class Not Found Class Not Found 1002 |  |

3

|  |  |  |  |
| --- | --- | --- | --- |
| 20 | 10 | Class Not Found Class Not Found 20 |  |

3

|  |  |  |  |
| --- | --- | --- | --- |
| 110  110  10 | Compile time error 'too many static blocks' | Compile time error 'Ambiguous declaration of i' | 110  110  110 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1,2 | 2,3 | 3,4 | 4,5 | 3 |
| Prints: 40 | Prints: 50 | Run time error |  | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| @annotation | @annotate | @meta | @metadata | 1 |
| 2,3,4 | 3,4,5 | 1,2,5 | 1,3,4 | 1 |
| java.lang.Annotat ion | java.annotation.A nnotation | There is no super type for an Annotation type. An Annotation type is just an interface defintion |  | 1 |
| Methods | Instance Variables | Local Variables | Classes | 1 |
| ElementType.AN NOTATION\_TYPE | ElementType.CLA SS | ElementType.INTE RFACE | ElementType.ENU M | 1 |
| 1,2,3 | 2,3,4 | 1,4,5 | 2,3,5 | 1 |
| An annotation can be declared at public, private, protected and package access levels | Annotation methods allow wrapper types as its return types, as an alternative to its equivalent primitive types | Annotations can replace interfaces in Java |  | 1 |
| Interfaces | Enums | Classes |  | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1,2,3 | 2,3,4 | 3,4,5 | 2,4,5 | 2 |
| @Target | @Accessible | @Runtime | @Reflexive | 2 |
| @Element | @ElementType | @Use |  | 2 |
| 1,2 | 3,4 | 4,5 | 1,5 | 2 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2,3 | 3,4 | 4,5 | 2,5 | 2 |
| 2,3 | 3,4 | 4,5 | 1,5 | 2 |
| The @Override can be used only while overriding a method declared in an interface | The @Override can be used only while overriding an abstract method in the super class | The @Override can be used only while overriding any built-in functions of Java Core API |  | 2 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| @Retention(Rete ntionPolicy.REFLE XIVE)  @Target(Element Type.METHOD)  public @interface PreInit {  int initValue() default 0;  } | @Retention(Rete ntionPolicy.RUNTI ME)  @Target(Element Type.METHOD)  public @interface PreInit {  int initValue() = 0;  } | @Retention(Reten tionPolicy.REFLEXI VE)  @Target(ElementT ype.METHOD)  public @interface PreInit {  int initValue() = 0;  } |  | 2  2  2 |
| 1,2 | 3,4 | 4,5 | 1,5 |
| Statement B is True and Statement A is false | Statement A is True and Statement B is false | Both the Statements A and B are true |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1-A, 2-B, 3-C, 4-D | 1-A, 3-B, 4-C, 2-D | 2-A, 4-B, 3-C, 1-D |  | 2 |
| ArrayList | Collection | List | Vector | 2 |
| 1,2 | 2,3 | 4,5 | 1,5 | 2 |
| Forces the user to mention the data type while declaring and creating instances of Generic Collections | Generic collections cannot be iterated using Iterator, only foreach statement works with Generic Collections | Use of generic collections removes thread safety for the collection class being used |  | 2 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1-A and B, 2-C, 3- E | 1-A and B, 2-C, 3- D | 1-A, 2-C, 3-D |  | 2 |
| Statement B is true and A is false | Both the statements A and B are true | Both the statements A and B are false. |  | 2 |
| Statement A is true and Statement B is false | Statement A is false and Statement B is true | Both the Statements A and B are true |  | 2 |
| Creates a Date object with '01-  01-1970 12:00:00  AM' as default value | Creates a Date object with 0 as default value | Creates a Date object with current date alone as default value |  | 2 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A,B | B,C | D,E | A,E | 2  2 |
| for (<String>Iterator myListIterator = myList.iterator(); myListIterator.ha sNext(); ) {  String myElement = myListIterator.ne xt();  System.out.printl n(myElement);  } | for (Iterator<String> myListIterator = myList.iterator(); myListIterator.Ne xt(); ) {  String myElement = myListIterator.ne xt();  System.out.printl n(myElement);  } | for (<String>Iterator myListIterator = myList.iterator(); myListIterator.has String(); ) {  String myElement = myListIterator.nex t();  System.out.println (myElement);  } |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Compilation error at line no 8 | Prints the output [Green World, 1, Green Peace] at line no 9 | Prints the output [Green World, Green Peace] at line no 9 |  | 2 |
| 3, 5, 7, 4 | 7, 5, 4, 3 | 3, 5, 7, 7, 4 |  | 2 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| {ee=ff} | {aa=bb, cc=dd, ee=ff} | {aa=bb, cc=dd} |  | 2 |
| Compilation error at the line commented as Line 4 | Runtime error at the line commented as Line 3 | Prints the output as  [Alaska, Fashion] [Alaska, Fashion] |  | 2 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Both Statements A and B are true | Statement A is false and Statement B is true | Both Statements A and B are false |  | 2  2  2 |
| B | C | A | E |
| [GSLV, PSLV, SLV,  Chandrayaan] | [GSLV, PSLV, IRS] | [GSLV, SLV,  Chandrayaan] |  |

2

|  |  |  |  |
| --- | --- | --- | --- |
| String dateFormat = "dd/mm/yyy hh:mm:ss t"; | String dateFormat  = "dd/MM/yyy hh:mm:ss t"; | String dateFormat  = "dd/MM/yyy hh:MM:ss a"; |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| String timeFormat = "hh:mm:ss t"; | String timeFormat  = "HH:MM:SS A"; | String timeFormat  = "HH:MM:SS T"; |  | 2 |
| HashSet | Hashtable | TreeSet |  | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| HashSet | ArrayList | LinkedList |  | 3 |
| HashSet | Stack | HashMap | ArrayList | 3 |
| Enumerator | Enumeration | Remover | Comparator | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Enumerator | Iterator | Enumeration | Browser | 3  3  3 |
| java.util.Calendar | java.util.Date and java.util.Time | java.util.Timestam p |  |
| java.util.Time | java.util.Date | java.util.Timestam p |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| public List<VariableObje ct> getRows(); | public List<Object> getRows(); | public List<String> getRows(); |  | 3 |
| private Map<Account, Set<User>> users  = null; | private Map<User, List<Account>> users = null; | private Map<Account, List<User>> users  = null; |  | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| public List<VariableObje ct> getRecords(); | public Set<Map<String, Object>> getRecords(); | public List<String> getRecords(); | public Set<Object> getRecords(); | 3 |
| 2,3 | 3,4 | 4,5 | 1,5 | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| List<Double> elements = new ArrayList<Double  >(); | List<Byte> elements = new ArrayList<Byte>(); | List<Short> elements = new ArrayList<Short>(); | List<Integer> elements = new ArrayList<Integer>()  ; | 3 |
| 2,3 | 3,4 | 4,5 | 1,5 | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prints: 1 0 | Prints: 2 0 | Prints: 2 -1 |  | 3 |
| [1, 3, 2, 1] | [1, 3, 1, 3, 1] | [1, 3, 2] |  | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1,2 | 3,4 | 4,5 | 3,4 | 3  3 |
| Compile time error at line number 09 | Prints the following without any error  Windows Linux Mac OS | Prints the following Windows Mac OS |  |

3

|  |  |  |  |
| --- | --- | --- | --- |
| Prints:  Chennai Delhi Pune Kolkata | Prints:  Chennai Pune Mumbai Kolkata | Compile Time Exception | Prints:  Chennai Delhi Mumbai Kolkata |

3

|  |  |  |  |
| --- | --- | --- | --- |
| LINE 1:  cal.add(Calendar. DAY\_OF\_MONTH  , -120);  LINE 2:  cal.add(Calendar. DAY\_OF\_MONTH  , 120); | LINE 1:  cal.subtract(Calen dar.DAY\_OF\_MO NTH, 120);  LINE 2:  cal.add(Calendar. DAY\_OF\_MONTH, 240); | LINE 1:  cal.subtract(Calen dar.DAY\_OF\_MON TH, 120);  LINE 2:  cal.add(Calendar.D AY\_OF\_MONTH, 120); |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1,2 | 2,3 | 3,4 | 3,5 | 3 |
| the value of i is 0 | the value of i is 1 | Compilation error | Runtime error | 2 |
| 2,3,4 | 3,4,5 | 1,4,5 | 1,2,3 | 2 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| byteValue() | intValue() | isInfinite() |  | 2 |
| 2,3 | 3,4 | 3,5 | 1,5 | 2 |
| 1,2,3 | 2,3,4 | 1,4,5 | 1,3,4 | 2 |
| while loop | do ... while loop | for loop | a loop formed using if and break | 2 |
| 2 4 8 16 32 | 2 4 8 1 6 | None of the listed options |  | 2 |
| break | continue | goto |  | 2 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2,4,5 | 1,2,3 | 2,3,4 | 3,4,5 | 2 |
| 1,2 | 3,4 | 4,5 | 1,4 | 2 |
| new Byte("10"); | new Long("3465"); | new Integer("637"); | new Boolean("truth"); | 2 |
| old truth  new truth two | new truth one new truth two | old truth  new truth one |  | 2 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Result: 100 | Result: 100  End of getSquare | Runtime Error | End of getSquare Result: 100 | 2 |
| 2,3,4 | 3,4,5 | 1,4,5 | 1,3,4 | 2 |
| Auto-boxing: 4,  Auto-Unboxing: 3 | Auto-boxing: 1,  Auto-Unboxing: 0 | Auto-boxing: 2,  Auto-Unboxing: 0 |  | 2 |
| for-each loop is an alternative to Enumeration | for-each loop does the automatic typecasting | for-each loop is an alternative to Iterator | for-each loop works with arrays | 2 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| The collection should be java.util.List type | The variable used for iterating should of java.lang.String type | The collection object should be delcared as final |  | 2 |
| 1,2 | 2,3 | 3,4 | 1,5 | 2 |
| 1411  java.lang.Number | 1411  java.lang.Object | Compile time error 'Cannot assign an integer value to Object' | Runtime error 'Unable to convert a long value to Object type' | 2 |

3

|  |  |  |  |
| --- | --- | --- | --- |
| 1,2 | 2,3 | 4,5 | 1,5 |

3

|  |  |  |  |
| --- | --- | --- | --- |
| 1,2 | 3,4 | 4,5 | 1,5 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Keeps on printing "Save Our Tigers" infinitely | Prints:  Save Our Tigers | Shows compile time error that labeled break cannot be used with switch statement |  | 3 |
| true false | false true | true true | Compile time error 'Increment/Decrem ent operator cannot be used with wrapper types' | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Runtime Error | Var-args | Long Wrapper | Compile-time Error | 3  3 |
| GB | RGB | No output |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| These kind of problems cannot be solved using if- statement | Inserting a break statement at the else part of every if-statement except the outermost if- statement will make the program to correctly calculate the grade | No changes required. The program correctly calculates the grade |  | 3 |
| 000012 | 0012 | 0000001112 |  | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2,3 | 1,4 | 3,5 | 2,5 | 3  3 |
| Prints: eTgr | After printing aeTgr throws StringIndexOutOf BoundsException | After printing eTgr throws StringIndexOutOfB oundsException |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| All sub class types of exceptions that come under java.lang.Runtim eException, if not handled inside a method should be declared using throws keyword | All sub class types of errors that come under java.lang.Error, if not handled inside a method should be declared using throws keyword | All sub class types of java.lang.Throwabl e, if not handled inside a method should be declared using throws keyword | All sub class types of exceptions that come under java.lang.Exception, if not handled inside a method should be declared using throws keyword | 2 |
| When a method in the super class is declared to throw a Checked Exception, the overriding method in the sub class should also declare the same. | The overriding methods cannot declare to throw the Super Class types of those exceptions declared in the super class methods. | the overriding method cannot re- declare the Unchecked exceptions, that are declared by super class method. |  | 2 |
| java.lang.Excepti on | java.lang.Error | java.lang.Throwabl e |  | 2 |
| 2,3 | 3,4 | 4,5 | 1,5 | 2 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1,2 | 2,3 | 3,4 | 1,4 | 2 |
| 1,2,3 | 2,3,4 | 3,4,5 | 1,4,5 | 2 |
| 1,2 | 2,3 | 4,5 | 3,5 | 2 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1,2,3 | 2,3,4 | 3,4,5 | 1,3,4 | 2 |
| 1,2 | 3,4 | 4,5 | 1,5 | 2 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prints "Previous Exception" | Prints "Last Exception" | Run time Error |  | 3 |
| Compiler time error Userdefined exceptions should extend Exception | Compile time error  Cannot use Throwable to catch the exception | Run time error test() method does not throw a Throwable instance |  | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Compile time error  Error class cannot be extended | Compile time error  Cannot catch Error type objects | Run time error test() method does not throw an Error type instance |  | 3 |
| IndexOutOfBoun dsException | NullPointerExcept ion | Compile time errors |  | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Shows unhandled exception type IOException at line number 5 | Demands a finally block at line number 4 | Shows unhandled exception type IOException at line number 4 |  | 3  3  3 |
| Finally Outer Exception Outer | Exception Outer Finally Inner Finally Outer | Exception Outer Finally Outer | Finally Inner Finally Outer Exception Outer |
| Run time error | Prints "Problem found" | Runs with No output |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Run time Error | Compile Time Error | Executes without any output |  | 3 |
| Compile time error; the declaration does not match the throw statement | Compiles successfully. But throws runtime error while executing | Compile time error; built-in exceptions like FileNotFoundExce ption cannot be instantiated programmatically |  | 3 |

3

|  |  |  |  |
| --- | --- | --- | --- |
| 1,2 | 3,4 | 4,5 | 1,5 |

3

|  |  |  |  |
| --- | --- | --- | --- |
| public void behaviour() throws SQLException {  System.out.printl n("Behaviour Implemented");  } | public void behaviour() throws IOException, SQLException {  System.out.printl n("Behaviour Implemented");  } | public void behaviour() throws Exception {  System.out.println ("Behaviour Implemented");  } | public void behaviour() throws IOException {  System.out.println( "Behaviour Implemented");  } |

3

|  |  |  |  |
| --- | --- | --- | --- |
| 2,3,4 | 3,4,5 | 1,4,5 | 1,2,5 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prints: Test  NullPointerExcep tion caught | Prints:  Exception caught Test | Prints:  Exception caught NullPointerExcepti on caught |  | 3  3 |
| 1,2,3 | 2,3,4 | 3,4,5 | 1,4,5 |

3

|  |  |  |  |
| --- | --- | --- | --- |
| Prints: Problem in Universe Problem in Universe | Prints: Problem in PlanetX Problem in PlanetY | Prints: Problem in Universe |  |

3

|  |  |  |  |
| --- | --- | --- | --- |
| Prints:  Specifc Problem 1 | Prints:  Specifc Problem 2 | Prints:  Specifc Problem 1  Specifc Problem 2 | Runtime Error |

3

|  |  |  |  |
| --- | --- | --- | --- |
| 1,2 | 2,3 | 3,4 | 3,5 |

3

|  |  |  |  |
| --- | --- | --- | --- |
| Shows, Unreachable catch block for Exception, at line number 6 | Shows, cannot create anonymous user defined exception class, at line number 12 | Shows, RuntimeException at line number 5 |  |

3

|  |  |  |  |
| --- | --- | --- | --- |
| finally NullPointerExcep tion  finally IllegalArgumentE xception  finally IndexOutOfBoun dsException | IndexOutOfBound sException  finally IllegalArgumentEx ception  finally NullPointerExcept ion  finally | NullPointerExcepti on  finally IllegalArgumentEx ception  finally IndexOutOfBounds Exception  finally |  |

3

|  |  |  |  |
| --- | --- | --- | --- |
| IllegalArgumentE xception IndexOutOfBoun dsException NullPointerExcep tion RuntimeExceptio n | RuntimeExceptio n RuntimeExceptio n RuntimeExceptio n RuntimeExceptio n | RuntimeException |  |

3

|  |  |  |  |
| --- | --- | --- | --- |
| RuntimeExceptio n RuntimeExceptio n RuntimeExceptio n | RuntimeExceptio n RuntimeExceptio n | RuntimeException |  |

3

|  |  |  |  |
| --- | --- | --- | --- |
| 2,3 | 3,4 | 4,5 | 1,5 |

3

|  |  |  |  |
| --- | --- | --- | --- |
| finally method1 FileNotFoundExc eption | finally method1 | FileNotFoundExce ption  finally method1 |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Returning from inner try Returning from inner finally 200 | Returning from inner try  100 | Shows compile time error 'Unreachable code at line numbers 11 and 15' | Shows compile time error 'Unreachable code at line number 15' | 3 |
| Soft Reference | Weak Reference | Phantom Reference |  | 2 |
| testObject.finaliz e(); | System.gc(); | testObject = null; | testObject.delete(); | 2 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1,2 | 2,3 | 3,4 | 2,4 | 2 |
| 1,2 | 2,3 | 3,5 | 4,5 | 2 |
| Weak Reference | Phantom Reference | Strong Reference |  | 2 |
| 2,3 | 3,4 | 1,4 | 3,5 | 2 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| The exception will be thrown to JVM and the garbage collection (finalization) of that object terminates | The exception will be ignored, but the garbage collection (finalization) of that object will be completed | The exception will be thrown to JVM and the garbage collection (finalization) of that object will also be completed |  | 2 |
| PhantomReferen ce is also called as Strong Reference | Checked by the garbage collector before throwing OutOfMemoryErr or | PhantomReferenc e extends SoftReference |  | 2 |
| java.lang.ref.Wea kReference.Queu e | java.lang.ref.Dead References | java.lang.ref.Queu edReferences |  | 2 |
| 3 objects | 2 objects | 0 object |  | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 objects | 3 objects | 0 object |  | 3 |
| secondString = null; | firstString = null; secondString = null; | firstString = null; |  | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| stringBuffer = null; | stringBuilder = null; | stringBuffer = null; stringBuilder = null; |  | 3  3 |
| 1,2 | 3,4 | 1,4 | 3,5 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| System.free(); | Set the value of each int to null | System.gc(); |  | 3 |
| Line No. 09 | Line No. 12 | Line No. 11 |  | 3 |

3

|  |  |  |  |
| --- | --- | --- | --- |
| 1 object | 2 objects | 0 object |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| At the line commented as // LINE 2 | At the line commented as // LINE 3 | At the line commented as // LINE 1 | None of the objects becomes eligible for garbage collection in the above code | 3 |
| 2,3 | 3,4 | 4,5 | 1,4 | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Declaration 2,4 | Declaration 1,2,3 | Declaration 2,3,4 |  | 1 |
| Has-a relationships always rely on inheritance. | Has-a relationships always require at least two class types. | Has-a relationships always rely on polymorphism. |  | 1 |
| 10,20 | 20,10 | Compilation error at line no:9 |  | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Addition Method returns integer | Runtime error at line no 12 | Addition Method returns nothing |  | 1 |
| hello JAVA | hello | nullJAVA |  | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2,3 | 3,4 | 4,5 | 2,5 | 1 |
| public void getNum(double  d) { } | public float getNum() { return 4.0f; } | public double getNum(float d) { return 4.0d; } |  | 1 |
| Statement A is false and B is true | Statement A is true and B is false | Both Statements A and B are false |  | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1,2 | 2,3 | 3,4 |  | 2 |
| It will not compile because of the different input type in the parameter list. | It will not compile because of the different return type. | It will compile and it is overriding |  | 2 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| WritingBooksRea dingBooks | BooksBooksReadi ngWriting | BooksReadingWriti ngBooks |  | 2 |
| 100100 | 10,10 | Compilation error |  | 2 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| public class Manager implements Payroll extends Employee {  public void getSalary() { /\*do something\*/ }  } | public class Manager extends Payroll implements Employee  public void getSalary (){ /\*do something\*/ }  public void Payroll.getSalary()  { /\*do something\*/ }  } | public class Manager implements Info extends Employee  {  public void Employee.getSalar y(){ /\*do something\*/ }  public void getSalary (){ /\*do something\*/ }  } | public class Manager implements Payroll extends Employee {  public void getSalary (){ /\*do something\*/ }  public void Payroll.getSalary(){  /\*do something\*/ }  } | 2 |
| class Shyam implements Tree  { } | class Shyam { private BestFriend Tree; } | class Shyam extends Tree { } | class Shyam { private Tree<bestFriend> } | 2 |
| An exception occurs at runtime at line 10. | Compilation fails because of an error in line 3. | Compilation fails because of an error in line 9. | Compilation fails because of an error in line 10. | 2 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Statement A is true and B is false | Both the statements are true | Both the statements are false |  | 2  2  2  3 |
| TRUE | Fred | An exception occurs at runtime. |  |
| Runnable r = new Runnable(public void run() { }); | Runnable r = new Runnable { public void run(){}}; | System.out.println (new Runnable()  {public void run() {  }}); | System.out.println( new Runnable(public void run() {})); |
| Outer.Inner oi = new Inner(); | Outer o = new Outer();  Outer.Inner oi = o.new Outer.Inner(); | Inner oi = new Outer.Inner(); |  |

3

|  |  |  |  |
| --- | --- | --- | --- |
| x, y, z, a | x, y, z, a, b | z, a, b | a, b |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| blue circle blue line | blue line blue line | blue circle blue circle |  | 3 |
| 0 | Runtime Error at line no 7 | 10 |  | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| interface B,C,D | interface A,B,C | interface B,C,D |  | 3 |
| 11 | 2 | Runtime error |  | 3 |
| Compile time error at line no 4 | Runtime error at line no 5 | Compile time error at line no 4 |  | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Compile time error at line no 5 | Runtime error at line no 5 | Compile time error at line no 2 |  | 3 |
| xyz abc ijk abc | abc xyz ijk xyz | xyz ijk abc ijk |  | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Top, i1=1,i2=2 | Check, i1=1,i2=2 | Top, i1=null,i2=null |  | 3 |
| 200100200 | Compile time error at Line no 1 | Runtime error at Line no 1 |  | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 | 0 | 1 |  | 3 |
| Line no 2 | Line no 3 | Line no 1 |  | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| main,First | Runtime error at line no 2 | main,Second |  | 3 |
| RoseSunflower | JasmineLilly | RoseLilly |  | 3 |

3

|  |  |  |  |
| --- | --- | --- | --- |
| System.out.printl n(this.name); | System.out.printl n(OuterLevel.this. name); | System.out.println (super.name); |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Compile time error caused by input not declaring Exception | Runtime error caused by input not declaring Exception | Compile time error caused by protected constructor for PrintData |  | 3 |
| Smile is-a Happy and has-a joy | joy has-a Happy and Happy is-a Smile. | Happy is-a joy and has-a Smile. |  | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 7 | 8 | 6 |  | 3 |
| 2,3,4 | 3,4,5 | 1,3,5 | 1,3,4 | 1 |
| 1,2 | 2,3 | 3,4 | 3,5 | 1 |
| 1,2 | 2,3 | 3,5 | 1,5 | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2,3,4 | 3,4,5 | 1,4,5 | 2,3,4 | 1 |
| Java Development Kit (JDK) | Java Database Connectivity (JDBC) | Java Debugger |  | 1 |
| Holds the location of Java Extension Library | Holds the location of Core Java Class Library (Bootstrap classes) | Holds the location of Java Software |  | 1 |
| 1,2,3 | 3,4,5 | 2,3,4 | 1,4,5 | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1,2 | 3,4 | 4,5 | 1,5 | 2 |
| 2,3,4 | 3,4,5 | 1,4,5 | 1,2,4 | 2 |
| Project References | Run/Debug Settings | Resource | Java Compiler | 2 |
| 2,3,4 | 3,4,5 | 2,4,5 | 1,2,5 | 2 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| To maintain the uniform standard | Helps JVM to find and execute the classes | Helps Javadoc to build the Java Documentation easily |  | 2  2  2  2 |
| main() method can be called only by JVM | Throws a runtime error 'Ambiguous method main()' | Compiler throws a compilation error 'Too many main() methods' |  |
| 1,2 | 2,3 | 3,4 | 1,5 |
| Classes can be loaded at Runtime, but the name of the class with full package name should be given in the code at compile time. | Classes cannot be loaded at Runtime | Only class that is loaded at runtime is the class that contains the main() method |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Packages in the extension directory of JRE / JDK  User-defined packages and libraries  Java Class Library | Packages in the extension directory of JRE / JDK  Java Class Library User-defined packages and libraries | User-defined packages and libraries  Java Class Library packages in the extension directory of JRE / JDK |  | 2 |
| 1,2 | 2,3 | 4,5 | 2,5 | 2 |
| 1,2 | 3,4 | 4,5 | 1,5 | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2,3,4 | 3,4,5 | 1,4,5 | 1,2,5 | 3  3 |
| FileProperties class | Properties class | System.properites |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2,3 | 3,4 | 4,5 | 1,5 | 3 |
| 1,2,3 | 2,3,4 | 1,4,5 | 1,2,4 | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1,2 | 3,4 | 4,5 | 1,5 | 3 |
| FileReader | DataInputStream and FileInputStream | FileInputStream | LookupStream | 3 |

3

|  |  |  |  |
| --- | --- | --- | --- |
| Product | Cart | UserInfo | Customer |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Task II | Task I | Task IV | Task V | 3 |
| BufferedReader and BufferedWriter | FileReader and FileWriter | BufferedInputStre am and BufferedOutputStr eam | FileInputStream and FileOutputStream | 3 |

3

|  |  |  |  |
| --- | --- | --- | --- |
| 1,2,3 | 3,4,5 | 1,4,5 | 2,4,5 |

3

|  |  |  |  |
| --- | --- | --- | --- |
| 1,2 | 2,3 | 3,4 | 4,5 |

3

|  |  |  |  |
| --- | --- | --- | --- |
| import java.io.Serializabl e;  class CardInfo { public String  cardNo; public String  cvvNo;  }  class CustomerInfo implements Serializable {  public String customerId;  public String customerName;  public CardInfo cardInfo;  } | import java.io.Serializabl e;  class CustomerInfo implements Serializable {  public String customerId;  public String customerName; public volatile  String cardNo; public volatile  String cvvNo;  } | import java.io.Serializable;  interface CardInfo  {  public String cardNo();  public String cvvNo();  }  class CustomerInfo implements CardInfo, Serializable {  public String customerId;  public String customerName; public String cardNo() { return  null; }  public String cvvNo() { return null; }  } | import java.io.Serializable;  interface CardInfo extends Serializable  {  public String cardNo();  public String cvvNo();  }  class CustomerInfo implements CardInfo {  public String customerId;  public String customerName; public String cardNo() { return  null; }  public String cvvNo() { return null; }  } |

3

|  |  |  |  |
| --- | --- | --- | --- |
| No errors; program compiles and executes properly.  Appends the text "Next Info" at the end of the file 'C:/TestRandom.t xt' | Runtime Error 'IOException: use shared more to open the same file more than once' | Compile time error 'Unhandled Exception FileNotFoundExce ption' |  |

3

|  |  |  |  |
| --- | --- | --- | --- |
| Compile time error 'Unable to resolve the method read(byte[], int, int) | Runtime error 'Invalid file open mode' | Compiles and executes properly. Reads and prints the last 7 bytes from the file 'C:/TestRandom.tx t' |  |

3

|  |  |  |  |
| --- | --- | --- | --- |
| 1,2 | 3,4 | 4,5 | 1,5 |

3

|  |  |  |  |
| --- | --- | --- | --- |
| Throws runtime error while reading the object 'Not a Serialized Object' | Throws compile time error 'Unsupported Serializable object' | No errors in the program. But fails to Serialize the object. So blank space is printed |  |

3

|  |  |  |  |
| --- | --- | --- | --- |
| class ReadOnlyFilter implements FileFilter {  public boolean accept(File file) {  if(file.isReadOnly(  ))  return true; else  return false;  }  } | class ReadOnlyFilter implements FileFilter {  public boolean accept(File file) {  if(file.canRead() && file.canWrite())  return true; else  return false;  }  } | class ReadOnlyFilter implements FileFilter {  public boolean accept(File file) {  if(file.canRead())  return true; else  return false;  }  } |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prints all names of files and directory that have write permission. | Prints all names of files and directory that starts with the letter 'A'. | Prints all names of files and directory that have write permission and that starts with the letter 'A'. |  | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Run time error | Compile time error | No errors in the program.  Compiles and executes successfully. Creates a text file 'TestData.txt' with the String array content in it. |  | 3 |
| 1,2 | 2,3 | 3,4 |  | 1 |
| Type 2 | Type 1 | Type 4 |  | 1 |
| Type 2 driver | Type 3 driver | Type 1 driver |  | 1 |
| 1 | 0 | null |  | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1,2,3 | 2,3,4 | 1,4,5 | 1,3,4 | 1 |
| getResultSetMet aData() | getMetaInfo() | getResultSetMetaI nfo() |  | 1 |
| getDatabaseMet aData() | getDBMetaData() | getDatabaseMetaI nfo() |  | 1 |
| DriverManager | Connection | ResultSet |  | 2 |
| java.sql.Date | java.sql.Time | java.util.Date |  | 2 |
| 1,2 | 3,4 | 4,5 | 1,5 | 2 |
| ResultSetMetaDa ta | ResultSet | RowSet |  | 2 |
| All IN parameteres must be registered with CallableStatemen t object prior to the execution | All INOUT  parameteres must be registered with CallableStatemen t object prior to the execution | No registration is required for any type of parameter |  | 2 |
| CONCUR\_UPDAT ABLE | CONCUR\_HOLDA BLE | No concurrency is associated by default |  | 2 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| The setter methods are used to set the values for the new row | The register methods are used to set the values for the new row | The values should be passed as parameters to the insertRow() method |  | 2  2  2  2 |
| 2,3 | 1,3 | 3.5 | 4,5 |
| ResultSetMetaDa ta | Connection | Driver |  |
| Starts a new Transaction and executes all the SQL statements, under that single transaction | Executes all SQL Statements without creating any transaction | There is no relation between transactions and auto-commit feature |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1,2 | 2,3 | 3,4 | 4,5 | 2  2  2 |
| Since the table gets created at the line number 1 itself, the s.executeUpdate(  ) in the line number 2 will throw SQLException | Tables should be created using createTable() method in the Connection interface | DDL statements can be executed only using Statement interface |  |
| Returns true if the column is a Primary Key, otherwise false | Returns true if the column is of any Number Type, otherwise false | Returns true only if the column has an index created on it |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| The query 'Select  \* from tab' can be used to get the list of tables | isTableAvailable() method in DatabaseMetaDat a can be used | Querying a table using Statement or PreparedStatemen t throws SQLException, if the table is not available. Thus it can be decided that the table does not exist |  | 2  2  2  2 |
| 1,2,3 | 3,4,5 | 1,4,5 | 1,2,5 |
| Throws an SQLException stating the error | Returns an Object array that contains the values of all matching columns | There cannot be more than one column with the same name in a Query as well as in the ResultSet |  |
| The ResultSet object remains as disconnected ResultSet with the previous results | The ResultSet object gets updated with the latest changes in the data | The data in ResultSet object becomes Read- only |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SQLException is thrown at the line number 2 | SQLException is thrown only at the time of processing the ResultSet object rs | PreparedStatemen t cannot detect the non-existence of table |  | 2  2  2 |
| 1,2,3 | 3,4,5 | 1,3,5 | 2,3,4 |
| System value Main value Face value | Face value Main value System value | Main Value Face Value System Value |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 2 3 4 | 1 3 4 2 | 1 2 3 4 5 | 2 3 4 5 | 2  2  2  2 |
| Only B is true | Only A is true | Both A and B are false |  |
| Prints the output "The Value of byte b is:130 | Compile time error at line number 4 | Compiles successfully but throws runtime error at line no 4 |  |
| bcdefgh | abcde | bcdef |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| The above code prints abc,def,hij 1,2,3 | The above code generates a Runtime error at line no 4 | The above code generates a Runtime error at line no 6 |  | 2 |
| Statement A is true and B is false | Statement A and B are true | Both Statements A and B are false |  | 2 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| createOutputFile(  )  deleteOutputFile(  )  testSomethingWi thFile() | testSomethingWit hFile() createOutputFile(  )  deleteOutputFile(  ) | deleteOutputFile() createOutputFile() testSomethingWit hFile() |  | 2 |
| Runtime Error | truetruefalse | truetruetrue |  | 2 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| The Person Name is Cameron Jimmy | The Person Name is Jimmy | The Person Name is female Cameron Jimmy |  | 2 |
| A,C,D is true and B is false | A,B is true and C,D is false | A,B,C,D is true |  | 2 |
| Program successfully executes and prints “value of c is 2 | Program compiles and but throws a runtime error | Program successfully executes but prints “value of c is ooxxoe |  | 2 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Statements B,C and D are correct | Statement B and C are correct | Statement A is correct |  | 2 |
| null,1,a | 0,1,a | Runtime error at line no 9 |  | 2 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ms.Jones Mr.smith | Mrs.Jones Ms.Smith | Mr.Jones Mrs.Smith |  | 2 |
| Pink Lotus Lilly | Pink White Violet | Compilation Error at line no 10 |  | 2 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| GoodMorning Noon  Evening Night | GoodMorning GoodAfterNoon GoodEvening GoodNight | Morning Noon Evening Night |  | 2 |
| Statement A is true and B is false | Statement A is false and B is true | Both Statements A and B are false |  | 2 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Integer.MAX\_VAL UE | 0 | None of listed options |  | 2 |
| 3,4 | Compile time error at line no 10 | Compile time error at line no 5 |  | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Runtime Error | Compile time error | Infinity  -Infinity Infinity | 1.4e99f  2.4e99f  3.4e99 | 3  3 |
| 2,2,2 | 0.2,0.5,0.5 | 1,2,2 |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 15,6 | 15,9 | Runtime error |  | 3 |
| A and C are correct | A and D are correct | A and B are correct |  | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Value Returned from the method is:10 | Value Returned from the method is:14 | Runtime error at line number 5 |  | 3 |
| The code prints out 'car'. | The code fails at runtime. | The code prints out 'Four Wheeler'. |  | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| class Play cannot access the enum constant FootBall without instantiation. | An Enum can not define an abstract method. | If the enum 'Games' is defined as an abstract 'enum', it will compile successfully. |  | 3 |
| A is true and B is false | Both A and B are true | Both A and B are false |  | 3 |
| It declares values to be a reference to an array object, but initializes it to null. | It declares values to be a reference to an array object which does not yet exist,but will contain 10 zeros when it does. | It declares values to be a reference to an array which contains 10 references to int variables. |  | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 18,10 | 5,9 | Compile time error at line no 11 |  | 3 |
| 1,2,null | Compile time error at line number 4 | 1,2 |  | 3 |
| for ( int j = names.length; j < names.length; j++  )  if ( names[j] != null )  System.out.printl n( names[j] ); | for ( int j = 0; j < names.length; j++  )  if ( names[j] != null )  System.out.printl n( names[j] ); | for ( int j = names.length; j >= 0; j++ )  if ( names[j] != null )  System.out.println ( names[j] ); |  | 3 |

3

|  |  |  |  |
| --- | --- | --- | --- |
| System.out.printl n (Personality.EXPR ESSIVE instanceof Personality); | System.out.printl n (EXPRESSIVE  instanceof Personality); | None of the listed options |  |

3

|  |  |  |  |
| --- | --- | --- | --- |
| POOR  Performance:Rev ised Salary = 100.0  AVERAGE  Performance:Rev ised Salary = 105.0  GOOD  Performance:Rev ised Salary = 125.0  EXCELLENT  Performance:Rev ised Salary = 140.0 | The above code will generate Compilation Error as enum cannot be applied for switch case | The constructor of an enum cannot accept 2 method parameters |  |

3

|  |  |  |  |
| --- | --- | --- | --- |
| Ruppee Blue  Doller Yellow | Ruppee Red  Doller Blue | Ruppee Blue  Doller Red  Dinar Yellow |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 20,10 | 10,20 | 5,10 |  | 3 |
| Garden status is Garden limit on the edge | Garden status is Flowers count OK | Error in line number 6 |  | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| short:1..99 byte:1..99 | short:- 32,768..32,767 byte:-128..127 | short:- 2,147,483,648  ..2,147,483,647  byte:1..99 |  | 3 |
| Code compiles successfully and executes successfully | Code compiles successfully; generates runtime error at line no 8 | Code compiles successfully; generates runtime error at line no 9 |  | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1,2 | 3,4 | 4,5 | 3,5 | 1 |
| 1632 | 1616 | 1734 |  | 1 |
| Comparing content of two objects | Comparing object Ids of two objects | Comparing creation time of two objects |  | 1 |
| Ensuring the memory allocation for the objects being created | Ensuring the object creation itself | Ensuring the objects added to JVM's Object pool |  | 1 |
| Line 2 | Line 3 | None of the given line uses JVM's Object Pool |  | 1 |
| 2,3 | 3,4 | 4,5 | 1,5 | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| IllegalArgumentE xception | ParseException | ArithmeticExceptio n |  | 1 |
| 1,2 | 3,4 | 4,5 | 3,5 | 1 |
| 1,2 | 3,4 | 3,5 | 1,5 | 1 |
| concat() | replace() | substring() |  | 2 |
| join() | trim() | toString() |  | 2 |
| Code Snippet 2 uses more JVM memory than Code Snippet 1 for storage | Both the Code Snippets uses same amount of memory | The usage of JVM's memory differs from JVM implementation |  | 2 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2,3 | 3,4 | 3,5 | 2,4 | 2  2  2  2  2 |
| 21 | 39 | 16 |  |
| Throws ParseException at line 2 | Throws IllegalArgumentEx ception at line 2 | Prints null |  |
| Throws NullPointerExcep tion at line 3 | Throws IllegalArgumentEx ception at line 2 | Prints false |  |
| symmetry | transitivity | reflexivity |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| The equals() method compares the hash code of current object and object to be compared, by calling hashCode() method and returns true if their hash code matches, and false otherwise. | There is no connection between equals() method and hashCode() method in the Object class. It simply compares only the content of both the object by just calling equals() method on the current object by passing the object to be compared. | The equals() method uses an object comparison algorithm which takes two different hash codes and evaluates to either true or false. The same is returned by equals() method. |  | 2 |
| Either equals() or hashCode() method can be overriden leaving the other as optional | Overriding of hashCode() method is not required. The implementation of hashCode() method in the Object class itself will serve all the purposes | Overriding of equals() method is not required. The implementation of equals() method in the Object class itself will serve all the purposes |  | 2 |
| PlayWithNuts | WithPeaNuts | PlayWithPea |  | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PlayWithNuts | PlayWithPea | PeaWithPlay |  | 3 |
| The thought is Green, | Green, so is the world | so is the World |  | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 9 | 3 | 5 |  | 3 |
| 8 | 10 | 11 |  | 3 |
| 9 | 8 | 7 |  | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| public boolean equals(MyClass that) {  if(that != null)  {  return (this.a == that.a) && (this.b  == that.b);  }  return false;  } | public boolean equals(Object obj1, Object obj2)  {  if(obj1 != null && obj2 != null && obj1 instanceof MyClass  && obj2 instanceof MyClass) {  MyClass this  = (MyClass) obj1; MyClass that  = (MyClass) obj2; return (this.a == that.a) && (this.b  == that.b);  }  return false;  } | None of the listed implemention is valid |  | 3 |
| public int hashCode() {  return super.hashCode;  } | public int hashCode() {  return this.hashCode;  } | Implementation of hashCode() is not required. Set is capable of finding duplicates by default |  | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| public int hashCode() {  return (int) System.currentTi meMillis();  } | public int hashCode() {  return super.hashCode();  } | public int hashCode() {  return super.hashCode()  +  name.hashCode();  } |  | 3  3 |
| public int hashCode() {  super.hashCode()  + 1000;  } | public int hashCode() {  return super.hashCode();  } | public int hashCode() {  return longitude.hashCod e() +  latitude.hashCode(  ) +  super.hashCode();  } | public int hashCode() {  return 1000+;  } |
| 2, 3 | 3, 1 | 1, 2 |  | 2 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Lock Starvation Condition | Race Condition | Lock Release Condition |  | 2 |
| notify() | notifyAll() | sleep() |  | 2 |
| Statement B is true and A is false | Statement A and B both are false | Statement A is true and B is false |  | 2 |
| 1,2 | 2,3 | 3,4 | 4,5 | 2 |
| Yes, because the methods are synchronized | Yes, two different threads hit | No, because the methods are synchronized |  | 2 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 | 3 | 4 | 1 | 2 |
| Statement B is true and A is false | Statement A and B both are false | Statement A is true and B is false |  | 2 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2,3 | 3,4 | 4,5 | 2,5 | 2 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Compiles successfully but generates no output | RunTime error | Prints I am RunningThread Three Times |  | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1,2 | 2,3 | 3,4 | 1,5 | 3 |
| Compilation error at Line no 2 | PlayPlayPlayPlay | Compilation Error at Line no 1 |  | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| start(); | Thread t = new Thread(this);  this.start(); | Thread t = new Thread();  this.start(t); |  | 3 |
| Prints 1 | Compile Time error at Line no 1 | Run Time error at Line no 1 |  | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 123 | 124 | Runtime time error IllegalMonitorStat eException when trying to wait at Line no 1 |  | 3 |
| 1,2 | 2,3 | 3,4 | 1,5 | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Statement A is False and B is True | Both Statements A and B are True | Statement A is True and B is False |  | 3 |
| 1,2,3 | 2,3,4 | 3,4,5 | 1,4,5 | 3 |
| A run time error indicating that no run method is defined for the Thread class | Clean compile and at run time the values 0 to 9 are printed out | A compile time error indicating that no run method is defined for the Thread class |  | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| synchronized(tan go){ tango.notifyAll();  } | synchronized(this  ){ this.notifyAll(); } | this.getCurrentThr ead().interrupt(); |  | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| NatureStarWars Motivates | Compilation error at Line no 1 | Runtime error at Line no 1 |  | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| HoneyBee | HoneyHoneyBee | Runtime Error |  | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prints Good and throws Runtime error | Prints an output as “GoodBest” | Compilation Error at Line no 2 |  | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Compilation and output of "Thunder Bolt" | Compilation and output of " Thunder Bolt 0 1  2 3" | Compile time error at Line no 1 |  | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| GreenPlanetGree nPlanetGreenPla net | StartingTree | Runtime error at Line no 1 |  | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prints the following output 2 times:  Guns  (waits for 1000 milli seconds)  Pistols | Prints MachineRunning and thows a Runtime Exception | Prints the following output 1 time:  Guns  (waits for 1000 milli seconds) Pistols |  | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| prints output: 0q | prints output: 1 | Compilation error at Line no 1 |  | 3 |
| 2,3 | 3,4 | 4,5 | 1,5 | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2,3 | 1,2 | 3,4 | 4,5 | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prints: AYZ | Prints: ABZ | Prints: XYZ |  | 3 |
| helloworld | Compilation Error at Line no 1 | worldhello |  | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2,3 | 3,4 | 4,5 | 1,5 | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prints CAT Three times and throws a Runtime exception | Prints CAT Six times | Prints CAT five times and throws a Runtime Exception |  | 3 |
| Statement A is False and B is True | Statement A is True and B is False | Both Statement A and B are False |  | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1,2 | 2,3 | 3,4 | 3,5 | 3 |
| 2,3 | 1,2 | 3,4 | 3,5 | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2,3 | 3,4 | 4,5 | 1,5 | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| new Thread(new Task()).start(); | new Thread(new Runnable(new Task())).start(); | new Thread(new Runnable(new RunnableTask())).s tart(); | This cannot be done, the class Task should implement Runnable | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| XYZ | Compilation Error at Line no 1 | Runtime Error at Line no 1 |  | 3 |
| Terminal | Output | Input | Keyboard | 1 |
| checkError() | printf() | format() |  | 1 |
| Kestrel | Merlin | Tiger | Playground | 1 |
| Image Morphing API for Java 3D | Image Processing API | Java Interpreter API |  | 1 |
| Annotations | Var-args | Generics | Iterable interface added as super interface for java.util.Collection | 1 |
| JAX | DOM | SAtX | SAX | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| HashSet | NavigableHashSet | NavigableTreeSet |  | 1 |
| java.util.concurre nt | java.util.logging | java.util.regex |  | 1 |
| Deque | LinkedDeque | BlockingDeque |  | 1 |