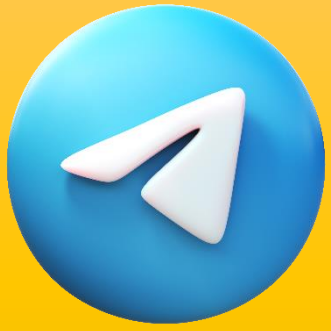
Explore More

Subcription : Premium CDAC NOTES & MATERIAL @99

Contact to Join
Premium Group



Click to Join
Telegram Group

For More E-Notes

Join Our Community to stay Updated

TAP ON THE ICONS TO JOIN!

	codewitharrays.in freelance project available to buy contact on 8007592194	
SR.NO	Project NAME	Technology
1	Online E-Learning Platform Hub	React+Springboot+MySql
2	PG Mates / RoomSharing / Flat Mates	React+Springboot+MySql
3	Tour and Travel management System	React+Springboot+MySql
4	Election commition of India (online Voting System)	React+Springboot+MySql
5	HomeRental Booking System	React+Springboot+MySql
6	Event Management System	React+Springboot+MySql
7	Hotel Management System	React+Springboot+MySql
8	Agriculture web Project	React+Springboot+MySql
9	AirLine Reservation System / Flight booking System	React+Springboot+MySql
10	E-commerce web Project	React+Springboot+MySql
11	Hospital Management System	React+Springboot+MySql
12	E-RTO Driving licence portal	React+Springboot+MySql
13	Transpotation Services portal	React+Springboot+MySql
14	Courier Services Portal / Courier Management System	React+Springboot+MySql
15	Online Food Delivery Portal	React+Springboot+MySql
16	Muncipal Corporation Management	React+Springboot+MySql
17	Gym Management System	React+Springboot+MySql
18	Bike/Car ental System Portal	React+Springboot+MySql
19	CharityDonation web project	React+Springboot+MySql
20	Movie Booking System	React+Springboot+MySql

	freelance_Project available to buy contact on 8007592194	
21	Job Portal web project	React+Springboot+MySql
22	LIC Insurance Portal	React+Springboot+MySql
23	Employee Management System	React+Springboot+MySql
24	Payroll Management System	React+Springboot+MySql
25	RealEstate Property Project	React+Springboot+MySql
26	Marriage Hall Booking Project	React+Springboot+MySql
27	Online Student Management portal	React+Springboot+MySql
28	Resturant management System	React+Springboot+MySql
29	Solar Management Project	React+Springboot+MySql
30	OneStepService LinkLabourContractor	React+Springboot+MySql
31	Vehical Service Center Portal	React+Springboot+MySql
32	E-wallet Banking Project	React+Springboot+MySql
33	Blogg Application Project	React+Springboot+MySql
34	Car Parking booking Project	React+Springboot+MySql
35	OLA Cab Booking Portal	React+NextJs+Springboot+MySql
36	Society management Portal	React+Springboot+MySql
37	E-College Portal	React+Springboot+MySql
38	FoodWaste Management Donate System	React+Springboot+MySql
39	Sports Ground Booking	React+Springboot+MySql
40	BloodBank mangement System	React+Springboot+MySql

41	Bus Tickit Booking Project	React+Springboot+MySql
42	Fruite Delivery Project	React+Springboot+MySql
43	Woodworks Bed Shop	React+Springboot+MySql
44	Online Dairy Product sell Project	React+Springboot+MySql
45	Online E-Pharma medicine sell Project	React+Springboot+MySql
46	FarmerMarketplace Web Project	React+Springboot+MySql
47	Online Cloth Store Project	React+Springboot+MySql
48	Train Ticket Booking Project	React+Springboot+MySql
49	Quizz Application Project	JSP+Springboot+MySql
50	Hotel Room Booking Project	React+Springboot+MySql
51	Online Crime Reporting Portal Project	React+Springboot+MySql
52	Online Child Adoption Portal Project	React+Springboot+MySql
53	online Pizza Delivery System Project	React+Springboot+MySql
54	Online Social Complaint Portal Project	React+Springboot+MySql
55	Electric Vehical management system Project	React+Springboot+MySql
56	Online mess / Tiffin management System Project	React+Springboot+MySql
57		React+Springboot+MySql
58		React+Springboot+MySql
59		React+Springboot+MySql
60		React+Springboot+MySql

Spring Boot + React JS + MySQL Project List

Sr.No	Project Name	YouTube Link
1	Online E-Learning Hub Platform Project	https://youtu.be/KMjyBaWmgzg?si=YckHuNzs7eC84-IW
2	PG Mate / Room sharing/Flat sharing	https://youtu.be/4P9clHg3wvk?si=4uEsi0962CG6Xodp
3	Tour and Travel System Project Version 1.0	https://youtu.be/-UHOBywHaP8?si=KHHfE_A0uv725f12
4	Marriage Hall Booking	https://youtu.be/VXz0kZQi5to?si=llOS-QG3TpAFP5k7
5	Ecommerce Shopping project	https://youtu.be/vJ_C6LkhrZ0?si=YhcBylSErvdn7paq
6	Bike Rental System Project	https://youtu.be/FIzsAmIBCbk?si=7ujQTJqEgkQ8ju2H
7	Multi-Restaurant management system	https://youtu.be/pvV-pM2Jf3s?si=PgvnT-yFc8ktrDxB
8	Hospital management system Project	https://youtu.be/IynIouBZvY4?si=CXzQs3BsRkjKhZCw
9	Municipal Corporation system Project	https://youtu.be/cVMx9NVyI4I?si=qX0oQt-GT-LR_5jF
10	Tour and Travel System Project version 2.0	https://youtu.be/_4u0mB9mHXE?si=gDiAhKBowi2gNUKZ

Sr.No	Project Name	YouTube Link
11	Tour and Travel System Project version 3.0	https://youtu.be/Dm7nOdpasWg?si=P_Lh2gcOFhlyudug
12	Gym Management system Project	https://youtu.be/J8_7Zrkg7ag?si=LcxV51ynfUB7OptX
13	Online Driving License system Project	https://youtu.be/3yRzsMs8TLE?si=JRI_z4FDx4Gmt7fn
14	Online Flight Booking system Project	https://youtu.be/m755rOwdk8U?si=HURvAY2VnizIyJlh
15	Employee management system project	https://youtu.be/lD1iE3W_GRw?si=Y_jv1xV_BljhrD0H
16	Online student school or college portal	https://youtu.be/4A25aEKfei0?si=RoVgZtxMk9TPdQvD
17	Online movie booking system project	https://youtu.be/Lfjv_U74SC4?si=fiDvrhhrjb4KSlSm
18	Online Pizza Delivery system project	https://youtu.be/Tp3izreZ458?si=8eWAOzA8SVdNwlyM
19	Online Crime Reporting system Project	https://youtu.be/0UlzReSk9tQ?si=6vN0e70TVY1GOwPO
20	Online Children Adoption Project	https://youtu.be/3T5HC2HKyT4?si=bntP78niYH802I7N

1) An abstract class in Java can be created using the keyword ____.

A) final

B) interface

C) abstract

D) static

Answer [=]

C

Explanation:
You should precede the keyword "abstract" before the keyword "class".

2) To create an Abstract class, the keyword "class" is also required. State TRUE or FALSE.

A) TRUE

B) FALSE

C) -

D) -

Answer [=]

A

Explanation:
True. You create a class using the keyword CLASS and make it abstract using the keyword ABSTRACT.

```
abstract class ClassA{ }
```


3) Can you create an object from an abstract class in Java?

A) Yes

B) No

C) -

D) -

Answer [=]

B

Explanation:
No. You can not instantiate or create an object from an abstract class.

4) Which is the opposite of an Abstract Class?

A) Interface

B) concrete class

C) -

D) -

Answer [=]

B

Explanation:
The Concrete Class is the opposite of an Abstract Class. Concrete class defines all the methods completely as opposed to abstract classes that leave implementation to the developers.

5) Choose a correct implementation of an Abstract class in the below Java code?

A)

```
abstract class ClassA { }
```

B)

```
abstract class ClassB
{
  abstract void method();
}
```

C)

```
abstract class ClassC
{
  void method()
  {
    System.out.print("Hello Abstract Class");
  }
}
```

D) All the above

Answer [=]

D

Explanation:
All the above. An abstract class need not contain a single abstract method also.

6) An abstract class in Java usually contains one or more abstract ____.

A) constructors

B) methods

C) variables

D) None

Answer [=]

B

Explanation:
Developers usually write abstract classes with one or more abstract methods to be implemented after inheriting the class.

7) Does the below Java code with abstract method compile?

```
class Puppy
{
  abstract void showName();
}
```

A) NO

B) YES

C) -

D) -

Answer [=]

A

Explanation:
No. An abstract method can be defined only by an abstract class.

8) What is the output of the below Java program with an abstract class?

```
abstract class Coffee
{
  Coffee()
  {
    System.out.println("Inside Constructor of Coffee..");
  }
}
class ColdCoffee extends Coffee
{
  ColdCoffee()
  {
    System.out.println("Inside Constructor of ColdCoffee..");
  }
}
public class AbstractClassTesting
{
  public static void main(String[] args)
  {
    ColdCoffee cf = new ColdCoffee();
  }
}
```

A) Compiler error

B)

```
Inside Constructor of Coffee..
Inside Constructor of ColdCoffee..
```

C)

```
Inside Constructor of ColdCoffee..
```

D)

```
Inside Constructor of Coffee..
```


Answer [=]

B

Explanation:
An abstract class can contain constructors. The order of invoking of constructors is from the Superclass to Subclass.

9) What is the output of the below Java program with an abstract class?

```
final abstract class Bell
{  }
class DoorBell extends Bell
{
  DoorBell()
  {
    System.out.println("DoorBell ringing..");
  }
}
public class AbstractClassTesting2
{
  public static void main(String[] args)
  {
    Bell bell = new DoorBell();
  }
}
```

A) DoorBell ringing..

B) No output

C) Compiler error

D) None of the above

Answer [=]

C

Explanation:
You can not use the keyword FINAL before an abstract class. The classes declared with the "final" keyword can not be subclassed or inherited. So, the FINAL keyword is banned for use with an abstract class. So, the compiler throws an error.

10) What is the output of the below Java program with an abstract class?

```
abstract class MathLibrary
{
  static final float PI = 3.1415f;
}

public class AbstractClassTesting3
{
  public static void main(String[] args)
  {
    System.out.println(MathLibrary.PI);
```

```
  }
}
```

A) No output

B) Compiler error

C) 3.1415

D) None of the above

Answer [=]

C

Explanation:
The output will be 3.1415. You can define and use a static final variable inside an abstract class. A static final variable is nothing but a constant. It is common to access a static variable with a DOT operator preceded by the Class Name.

11) What is the output of the below Java program with multiple abstract classes?

```
abstract class Editor
{
  abstract void show();
}

abstract class Author extends Editor
{
  abstract void print();
}

class Office extends Author
{
  void show()
  {
    System.out.println("Editor method");
  }
  void print()
  {
    System.out.println("Author method");
  }
}

public class AbstractClassTesting4
{
  public static void main(String[] args)
  {
    Editor ed = new Office();
```

```
    ed.show();

    Author au = new Office();
    au.print();
  }
}
```

A)

```
Editor method
Editor method
```

B)

```
Author method
Author method
```

C)

```
Editor method
Author method
```

D) Compiler error

Answer [=]

C

Explanation:
An abstract class can inherit from another abstract class. The final concrete class inheriting the last abstract in the chain should implement all the abstract methods above it.

12) An object of multi-level inherited abstract class can not be created in memory? State TRUE or FALSE.

A) TRUE

B) FALSE

C) -

D) -

Answer [=]

A

Explanation:

True. You can not create objects out of an abstract class whether it is single level or multilevel.

13) An abstract class with 100% abstract methods is equivalent to _____.

A) Concrete class

B) Virtual Class

C) Interface

D) All the above

Answer [=]

C

Explanation:
An interface contains only abstract methods. So it is comparable to a 100% abstract class.

14) What is the output of the below Java program with an abstract class?

```
public abstract class AbstractClassTest5
{
  public static void main(String[] args)
  {
    System.out.println("Inside Main() method..");
  }
}
```

A) No output

B) Compiler error

C) Inside Main() method..

D) None of the above

Answer [=]

C

Explanation:
It is perfectly allowed to define a public abstract class as long as you do not create objects of it. Remember that the name of the JAVA file is the name of the public class.

15) What is the output of the below Java code with an abstract class and inner class?

```
public abstract class AbstractClassTest6
{
  class Anonymous
  {
    int a=5;
  }
  public static void main(String args[])
  {
    System.out.print("Inner class is present..");
  }
}
```

A) Compiler error

B) Inner class is present..

C) No output

D) None of the above

Answer [=]

B

Explanation:
Inner classes are allowed to be written inside an abstract class. So, the program compiles successfully. But to use that inner class, subclassing is necessary.

16) Choose a correct statement about abstract classes?

A) An abstract class can extend a concrete class

B) An abstract class can extend another abstract class

C) An abstract class can implement any number of interfaces.

D) All the above.

Answer [=]

D

Explanation:
Yes. Abstract classes can implement interfaces and extend other classes.

17) Choose correct statements about an Abstract class in Java?

A) An abstract class implementing an Interface, need not implement methods of an interface

B) An abstract class extending another abstract class, need not define methods of the super abstract class.

C) The first subclass of an abstract class should define all the abstract methods inherited from all the interfaces and super abstract classes.

D) All the above

Answer [=]

D

18) Just like an Interface, you can define constants in abstract classes in Java. State TRUE or FALSE.

A) TRUE

B) FALSE

C) -

D) -

Answer [=]

A

Explanation:
TRUE.

19) Abstract classes support ____ inheritance.

A) Multiple

B) Multilevel

C) -

D) -

Answer [=]

B

Explanation:
Just like a concrete/normal class in Java, abstract classes support multilevel inheritance only. You can extend only from one super class.

20) An abstract class can define ____ inside of it.

A) Inner abstract class

B) Inner concrete class

C) static methods

D) All the above

Answer [=]

D

1) Which are access modifiers below in Java?

A) public

B) private

C) protected

D) All the above

Answer [=]

D

Explanation:
Java provides 4 access modifiers namely public, private, protected, and default(not a keyword).

2) What is an access modifier in Java?

A) An access modifier controls the visibility of variables, constants and methods of a class

B) An access modifier can hide or show a variable or method to outside classes.

C) Access modifiers help in implementing the Encapsulation feature of Object-Oriented Programming (OOPs).

D) All the above

Answer [=]

D

3) Which is the keyword used to specify the DEFAULT access modifier in java?

A) default

B) bydefault

C) normal

D) There is no keyword

Answer [=]

D

Explanation:
Yes. If you do not specify any access modifier before a variable or method, it is considered to be declared under the default access modifer.

4) Which is the less restrictive access modifier in Java?

A) public

B) private

C) protected

D) default

Answer [=]

A

Explanation:
A variable or a method marked PUBLIC is available to all outside classes irrespective of package that a class lives in. So, PUBLIC is the least restrictive access modifier in Java.

5) Which is the most restrictive access modifier in Java?

A) public

B) private

C) protected

D) default

Answer [=]

B

Explanation:
A variable or method marked PRIVATE is accessible only within the class. Outside classes can not access these. So, PRIVATE is the most restrictive access modifier in Java.

6) Is it possible to combine more than one access modifier in Java?

A) No

B) Yes

C) -

D) -

Answer [=]

A

Explanation:
No. A method, variable or class can be marked with only one access modifier.

7) Choose the correct Java code snippet with access modifiers below?

A)

```
class ABC
{

}
```

B)

```
private protected class ABC
{

}
```

C)

```
protected public class ABC
{

}
```

D)

```
protected private class ABC
{

}
```


Answer [=]

A

Explanation:
Class ABC has a default access modifier. It is not mentioned as it does not have a keyword. You can not combine more than one access modifiers in Java.

8) Can you access a default member of a class from a class of outside current package in Java?

A) No

B) Yes

C) -

D) -

Answer [=]

A

Explanation:
No. You can not access a default member from outside the package.

9) Choose the correct statement about access modifiers in Java.

A) public, protected and default variables and methods can be accessed outside the package somehow.

B) public and protected variables and methods can be accessed outside the package somehow.

C) Only public variables and methods can be accessed outside the package somehow.

D) None of the above

Answer [=]

B

10) To access a protected variable or method of a Class outside the package, you need to ____ in Java.

A) Create an instance and call the protected variable or method

B) Create a Subclass and call the protected variable or method

C) A and B

D) None of the above

Answer [=]

B

Explanation:
Protected variables and methods must be called from inside subclasses or subclass objects only.

11) What is the output of the Java program with access modifiers?

```
package apartment1;

public class Class101
{
  public String colour = "BLUE";
}


package apartment2;
import apartment1;
```

```
public class Apartment2Flat102
{
  public static void main(String[] args)
  {
    Class101 flat = new Class101();
    System.out.println(flat.colour);
  }
}
```

A) BLUE

B) empty

C) null

D) Compiler error

Answer [=]

D

Explanation:

```
You should import either a specific Class or all Classes of a Java Package.
So, use
import apartment1.*; or import apartment1.Class101;

error: Class101 cannot be resolved to a type
```

12) What is the output of the below Java Code Snippet with access modifiers?

```
package package1;

public class Forest {
int area = 10;//sq. km
}


package package1;

public class SmallForest
{
  public static void main(String[] args)
  {
    Forest fr = new Forest();
    System.out.println("Area = " + fr.area);
  }
}
```

A) Area = 0

B) Area =10

C) No ouput

D) Compiler error

Answer [=]

B

Explanation:
The DEFAULT variables or methods of class can be called inside any class of the same Package.

13) What is the output of the Java code snippet with default access modifier?

```
package package1;

public class Bug {
int legs = 4;
}


package package2;
import package1.Bug;

public class BugTest extends Bug
{
  public static void main(String[] args)
  {
    Bug bug = new Bug();
    System.out.println("Bug Legs = " + bug.legs);
  }
}
```


A) Bug Legs = 4

B) Bug Legs = 0

C) Bug Legs =

D) Compiler error

Answer [=]

D

Explanation:

You can not access a DEFAULT variable or method of a class from outside package in any means even through inheritance. So, the compiler throws error.

```
Unresolved compilation problem: The field Bug.legs is not visible
```

14) What is the output of the Java code snippet with PROTECTED access modifier?

```
package package1;

public class Parent
{
  protected void showProperty()
  {
    System.out.println("10 million");
  }
}


package package2;
import package1.*;

public class Children extends Parent
{
  void show()
  {
    showProperty();
  }
  public static void main(String[] args)
  {
    Children chil = new Children();
    chil.show();
  }
}
```


A) No output

B) 10 million

C) Compiler error

D) None of the above

Answer [=]

B

Explanation:

Through inheritance, you can access a PROTECTED variable or method of a class even from outside the package. Here, we accessed Parent Class of Package1 from Children Class of Package2.

15) What is the output of the Java code with PRIVATE access modifier?

```
package package1;
public class Factory
{
  private void run()
  {
    System.out.println("Running factory");
  }
}


package package1;
public class FactoryTest
{
  void callFactory()
  {
    Factory fac2 = new Factory();
    fac2.run();
  }
  public static void main(String[] args)
  {
    Factory fac1 = new Factory();
    fac.run();
  }
}
a
```


A)

```
Running factory
Running factory
```

B)

```
Running factory
```

C) Compile error

D) None of the above

Answer [=]

C

Explanation:
PRIVATE variables or methods of a Class are visible only within that Class. No other class can refer to these PRIVATE things even through inheritance. So, the compiler produces error.

```
Unresolved compilation problem: The method run() from the type Factory is not visible
```

16) In Java, a PROTECTED variable or method of a Super class can not be accessed from a STATIC method of Subclass of different PACKAGE. State TRUE or FALSE.

A) TRUE

B) FALSE

C) -

D) -

Answer [=]

A

17) What is the output of the Java program with PROTECTED access modifer?

```
package package1;

public class Milk
{
  protected int volume = 1000; //Liters
}

package package2;
import package1.Milk;
public class RoseMilk extends Milk
{
  public static void main(String[] args)
  {
    Milk object = new Milk();
    System.out.println(object.volume);
  }
}
```

A) 0

B) 1000

C) null

D) Compiler error

Answer [=]

D

Explanation:
You can access PROTECTED things from within the same package in either STATIC or NON-STATIC methods. But, if you want to call PROTECTED things from outside PACKAGE, you should move the code to a non-static method.

18) What is the output of the Java code with PROTECTED access modifier?

```
package package1;
public class Plant
{
  protected void showHeight()
  {
    System.out.println("50 meters");
  }
}


package package2;
import package1.Plant;

public class Tree extends Plant
{
  public static void main(String[] args)
  {
    Tree tree = new Tree();
    tree.showHeight();
  }
}
```


A) No output

B) 50 meters

C) Compiler error

D) No output

Answer [=]

B

Explanation:

Here, a PROTECTED method of super class has been accessed using Subclass reference. So, the compiler does not produce error even though, it has been called inside STATIC method. If you use Superclass reference or object to access PROTECTED things, you will get error.

1) Choose a correct statement about Autoboxing in Java?

A) Boxing is the process of converting a primitive data to its equivalent Wrapper class type

B) Unboxing is the process of converting a Wrapper class object to its equivalent primitive data type

C) Autoboxing is the process of converting Primitive data type to Wrapper Object type automatically. Auto-Unboxing is the process of converting Wrapper type object to Primitive type automatically.

D) All the above

Answer [=]

D

2) Which does autoboxing or unboxing in Java?

A) Compiler

B) Runtime

C) Third party tools

D) None

Answer [=]

A

Explanation:
Yes. The compiler does all the Autoboxing and unboxing for you.

3) Autoboxing or unboxing involves replacement of Wrapper objects with Primitives and vice versa. State TRUE or FALSE.

A) TRUE

B) FALSE

C) -

D) -

Answer [=]

A

4) Which Java version introduced the feature Autoboxing and Unboxing?

A) Java 4

B) Java 5

C) Java 6

D) Java 7

Answer [=]

B

Explanation:
Yes. Java 5 introduced many new features like AUTOBOXING, UNBOXING, advanced FOR loop, VARARGS, ENUM, GENERICS etc.

5) What is the need for Autoboxing and Unboxing in Java?

A) Reduces code to be written by developers to convert from Wrapper to Primitive vice versa

B) Compile time is reduced due to fixed methods followed to automatically convert Objects and Primitives

C) Auto-unboxing improves memory efficiency by converting Wrapper to Primitive and process when object version is not needed

D) All the above

Answer [=]

D

6) Choose the right statement which does autoboxing in Java?

A)

```
Integer temperature1 = 100;
```

B)

```
int temperature2 = 101;
```

C)

```
Integer temperature1 = 100;
int temperature2 = 101;
```

D) All the above

Answer [=]

A

Explanation:
As part of auto-boxing, an INT number is converted to INTEGER object automatically.

7) Choose the correct way of autoboxing Primitives to Wrapper objects below?

A)

```
Float f1 = 1.0f;
```

B)

```
Boolean bull = false;
```

C)

```
Character ch = 'a';
```

D) All the above

Answer [=]

D

8) Does the below Java code undergoes autoboxing correctly?

```
long weight = 10;
Long wei = weight;
System.out.println(ch.SIZE);
```

A) Yes

B) No

C) -

D) -

Answer [=]

A

Explanation:
Yes. You can assign a primitive LONG value to a Wrapper LONG object. The assignment undergoes auto-boxing by the compiler.

9) Auto-boxing or Auto-unboxing involves assignment (not always) in Java. State TRUE or FALSE.

A) TRUE

B) FALSE

C) -

D) -

Answer [=]

A

Explanation:
TRUE

10) Auto-boxing or Auto-unboxing does not work inside static methods in Java. State TRUE or FALSE

A) FALSE

B) TRUE

C) -

D) -

Answer [=]

A

Explanation:
False. The method type can be anything.

11) What is the output of the below java code with autoboxing?

```
public class AutoBoxingTest2
{
  static void show(int reading)
  {
    System.out.println("Reading: " + reading);
  }
  public static void main(String[] args)
  {
    Integer a = Integer.valueOf(10);
    show(a);
  }
}
```

A) Reading: 0

B) Reading: 10

C) Compiler error

D) None

Answer [=]

B

Explanation:
Autoboxing or unboxing works with method arguments or parameters.

12) Which is faster Auto-boxing or Auto-unboxing?

A) Auto-boxing

B) Auto-unboxing

C) -

D) -

Answer [=]

B

Explanation:
Auto-unboxing is faster as it does not need to allocate memory for creation of Object. An already existing object will be used to extract value instead. Auto-boxing involves creation of Object and assigning value.

13) Does autoboxing work with creation of an Array in Java?

A) Yes

B) No

C) -

D) -

Answer [=]

A

Explanation:
Yes. You can create an Array of Wrapper objects just like creating a Primitive array. Example is below.

```
Integer k = 20;
Integer ary[] = {1, 4, 8, k};
System.out.println(ary[3]);
```

```
//OUTPUT
20
```

14) Autoboxing or Auto-unboxing works with ___ in Java.

A) Loops, Switch statements

B) Ternary Operator

C) IF ELSE statements

D) All the above

Answer [=]

D

15) Autoboxing or Auto-unboxing is ___ in Java.

A) Implicit

B) Explicit

C) -

D) -

Answer [=]

A

Explanation:
Implicit.

16) Choose the correct code that does auto-unboxing?

A)

```
Character ch = 'S';
switch(ch)
{
  case 'S': System.out.println("OK"); break;
  case 'p': break;
}

//OUTPUT
OK
```

B)

```
Integer t = 90;
int abc = ++t;
System.out.println(abc);

//OUTPUT
91
```

C)

```
int b =  10;
Integer c = 20;
if(b < c)
{
  System.out.println("LESS");
}

//OUTPUT
LESS
```

D) All the above

Answer [=]

D

17) What is the output of the below Java code snippet that use Autoboxing and Unboxing?

```
public class AutoUnboxingTest1
{
  public static void main(String[] args)
  {
    int apple1 = 10;
    Integer apple2 = 10;
    if(apple1 == apple2)
      System.out.println("apple1=apple2");
```

```
    else
      System.out.println("apple1!=apple2");
  }
}
```

A)

```
apple1=apple2
```

B)

```
apple1!=apple2
```

C) Compiler error

D) Non

Answer [=]

A

Explanation:
Both primitive and wrapper versions hold the same number. So, the comparison results in equality.

18) Autoboxing and unboxing in Java language saves the programmer from errors and extra lines of code. State TRUE or FALSE.

A) TRUE

B) FALSE

C) -

D) -

Answer [=]

A

Explanation:
True. You can use either Primitive version or Wrapper object version interchangeably. The compiler or Runtime does not throw errors.

19) What is the output of the below Java code with Autoboxing and unboxing?

```
float f1 = 10.0f;
Float f2 = Float.valueOf(10);
```

```
if(f1 == f2)
  System.out.println("FLOATs are equal.");
else
  System.out.println("FLOATs are not equal.");
```

A) FLOATs are equal.

B) FLOATs are not equal.

C) Compiler error

D) None

Answer [=]

A

Explanation:
You can compare float and Float variables as these are automatically boxed and unboxed.

20) Choose the correct statement with auto-unboxing in Java?

A)

```
Short s1 = (short)30;
short s2 = s1;
```

B)

```
Boolean b1 = Boolean.valueOf(true);
boolean b2 = b1;
```

C)

```
Character ch1 = Character.valueOf('a');
char ch2 = ch1;
```

D) All the above

Answer [=]

D

1) Java Enum is also called ___.

A) Enumeration

B) Numeration

C) Iteration

D) None of the above

Answer [=]

A

Explanation:
Enumeration and Enum are one and the same in Java.

2) Which is the keyword used to create an ENUM type in Java?

A) enum

B) Enum

C) enumeration

D) Enumeration

Answer [=]

A

Explanation:
"enum". All the letters should be in lower case.

3) An ENUM type in Java is like a ____.

A) interface

B) class

C) abstract class

D) None of the above

Answer [=]

B

4) An ENUM class in Java can contain _____.

A) constructors

B) methods

C) variables and constants

D) All the above

Answer [=]

D

5) Which is true about enum constants in Java?

A) Enum constants have index.

B) The index starts with zero (0).

C) Enum constant can be accessed using a DOT (.) operator

D) All the above

Answer [=]

D

6) Java enums are introduced in which version?

A) Java 1.4

B) Java 1.5

C) Java 6

D) Java 8

Answer [=]

B

Explanation:
Java 1.5

7) An enum variable in java can take ___ as the assigned value in Java.

A) Any integer

B) Any non null object

C) Enum constant only

D) None of the above

Answer [=]

C

Explanation:
An enum variable can only hold enum constants.

8) The name of an ENUM or the name of ENUM constant can be ____ in Java.

A) A string of characters that starts with an Underscore, Dollar or Alphabet

B) A string of characters that can contain numbers at any position other than zero

C) A string of characters that can be an existing keyword

D) All the above

Answer [=]

D

9) What is the output of the below Java program with Enums?

```
enum Car
{
  SEDAN, HATCHBACK, SUV
}
public class EnumTest1
{
  public static void main(String[] args)
  {
    Car c1 = Car.SEDAN;
    System.out.println(c1.name() + ", " + c1);
  }
}
```

A) SEDAN, 0

B) SEDAN, SEDAN

C) SEDAN,

D) None of the above

Answer [=]

B

10) What is the output of the below Java program with Enums?

```
enum Bus
{
  DIESEL, ELECTRIC, HYBRID
  int mileage = 10;
}
public class EnumTest2
{
  public static void main(String[] args)
  {
    Bus bus = Bus.ELECTRIC;
```

```
    System.out.println(bus + ", " + bus.mileage);
  }
}
```

A) ELECTRIC, 0

B) ELECTRIC, 10

C) 1, 0

D) Compiler error

Answer [=]

D

Explanation:
If you want to add code other than enum-constants inside an ENUM class, the constants should end with a semicolon. Otherwise, the semicolon is not required.

```
enum Bus
{
  DIESEL, ELECTRIC, HYBRID; //semicolon
  int mileage = 10;
}
```


11) Can you define STATIC variables and STATIC methods inside an ENUM in Java?

A) Yes

B) No

C) -

D) -

Answer [=]

A

Explanation:
The STATIC variables and methods are common to all ENUM constants. There will be only one copy of the static variable shared by all enum-constants.

12) An ENUM can be defined in the place ____ in Java?

A) Inside a class but outside of methods

B) Outside a class or interface

C) Inside an Interface but outside methods (abstract if any)

D) All the above

Answer [=]

D

13) An ordinal() method of an ENUM returns _____.

A) Hash value

B) Enum constant length

C) Enum constant index which is zero based

D) None of the above

Answer [=]

C

Explanation:

```
enum Dog
{
  SMALL, MEDIUM, BIG;
}
Dog dg = Dog.SMALL;
System.out.println(dg.ordinal());
//OUTPUT
0
```


14) Which is the correct way of implementing ENUM constants inside SWITCH statement in Java below?

```
enum Marks
{
  AVERAGE, GOOD, EXCELLENT;
}
public class EnumTest3
{
  public static void main(String[] args)
  {
    Marks student2 = Marks.GOOD;

    //MISSING SWITCH STATEMENT
  }
}
```

A)

```
switch(student2)
{
case AVERAGE: System.out.println("Marks: 60"); break;
case GOOD: System.out.println("Marks: 80"); break;
case EXCELLENT: System.out.println("Marks: 90");
}
```

B)

```
switch(student2)
{
case Marks.AVERAGE: System.out.println("Marks: 60"); break;
case Marks.GOOD: System.out.println("Marks: 80"); break;
case Marks.EXCELLENT: System.out.println("Marks: 90");
}
```

C)

```
switch(student2)
{
case Marks.AVERAGE: System.out.println("Marks: 60"); break;
case GOOD: System.out.println("Marks: 80"); break;
case Marks.EXCELLENT: System.out.println("Marks: 90");
}
```

D)

```
switch(student2)
{
case Marks[0]: System.out.println("Marks: 60"); break;
case Marks[1]: System.out.println("Marks: 80"); break;
case Marks[2]: System.out.println("Marks: 90");
}
```


Answer [=]

A

Explanation:
The enum-constants should be used directly as switch-case constants without using enum-name.

15) Can you pass values to ENUM constructors in Java?

A) No

B) Yes

C) -

D) -

Answer [=]

A

Explanation:
NO. You can not pass values to enum-constructor from outside.

16) What is the output of the below Java code with Enums and Constructors?

```
enum Ship
{
  BOAT(5), SHIP(10), VESSEL(100);
  int capacity;
  Ship(int i)
  {
    capacity = i;
  }
  Ship(){}
  void show() {System.out.println("Capacity: " + capacity);}

}
public class EnumTest4
{
  public static void main(String[] args)
  {
    Ship s1 = Ship.BOAT;
    s1.show();
```

```
  }
}
```

A) Capacity: 5

B) Capacity: 0

C) Capacity: 10

D) Capacity: 100

Answer [=]

A

Explanation:
Each ENUM constant has a separate copy of the instance variable "capacity". The constructor is called 3 times as there are 3 enum-constants inside.

17) What is the output of the below Java code with constructors?

```
enum Human
{
  KID(10), BOY(30), MAN(60);
  Human(int weight)
  {
    System.out.println("Weight: " + weight);
  }
}
public class EnumTest5
{
  public static void main(String[] args)
  {
    Human he = Human.KID;
  }
}
```


A)

```
Weight: 10
Weight: 30
Weight: 60
```

B)

```
Weight: 10
Weight: 10
```

```
Weight: 10
```

C)

```
Weight: 0
Weight: 0
Weight: 0
```

D) No output

Answer [=]

A

Explanation:
If you try to create at least one Enum variable, the constructor is called for all enum-constants for the first time.

18) Which is the correct way of looping through all ENUM constants in Java below?

```
enum FRUIT
{
  ORANGE, PEARS, PLUMS;
}
```

A)

```
for(FRUIT fr2: FRUIT.values())
System.out.println(fr2);
```

B)

```
//new ENUM array
FRUIT ary[] = new FRUIT[3];
ary[0] = FRUIT.ORANGE;
ary[1] = FRUIT.PEARS;
ary[2] = FRUIT.PLUMS;
for(FRUIT fr: ary)
  System.out.println(fr);
```

C) Both A and B

D) None of the above

Answer [=]

C

Explanation:
You can call values() method to get an array of enum-constants. Here, you are using Advanced FOR Loop or FOR-EACH loop to loop through all constants.

19) An ENUM can implement an INTERFACE. State TRUE or FALSE.

A) TRUE

B) FALSE

C) -

D) -

Answer [=]

A

Explanation:
Yes. Example is below.

```
interface Jam
{
  void show();
}

enum FRUIT implements Jam
{
  ORANGE("Orange"), PEARS("Pears"), PLUMS("Plums");
  String text="";
  FRUIT(String fr){text=fr;}
  @Override
  public void show() {
    System.out.println(text + " Jam");
  }
}

public class EnumTest7
{
  public static void main(String[] args)
  {
    FRUIT fru = FRUIT.ORANGE;
    fru.show();
  }
}
```

```
//OUTPUT
Orange Jam
```

1) What is the output of the below Java code with exceptions?

```
public class ExceptionTest8
{
  public static void main(String[] args) throws ArithmeticException
  {
    System.out.println("I am happy.");
    throw new ArithmeticException();
  }
}
```

A) I am happy.

B) No output

C) Compiler error

D) None of the above

Answer [=]

A

Explanation:
ArithmeticException is an unchecked exception. So, the compiler does not stop with errors.

2) What is the output of the Java code with exceptions?

```
public class ExceptionTest9
{
  public static void main(String[] args)
  {
    System.out.println("I am going to forest.");
    throw new ClassNotFoundException();
  }
}
```

A) I am going to forest.

B) No output

C) Compiler error

D) None of the above

Answer [=]

C

Explanation:

ClassNotFoundException is a checked exception in Java. So, the compiler throws error for not handling it using TRY CATCH FINALLY statements.

```
Unresolved compilation problem: Unhandled exception type ClassNotFoundException
```

3) What is the output of the below Java code with exception handling keywords?

```
public class ExceptionTest10
{
  public static void main(String[] args) throws Exception
  {
    System.out.println("Good Morning Cortana.");
    throw new NoSuchMethodException();
  }
}
```

A) Good Morning Cortana.

B)

```
Good Morning Cortana.
Exception in thread "main" java.lang.NoSuchMethodException
        at Project1/package1.ExceptionTest10.main(ExceptionTest10.java:8)
```

C) Compiler error

D) None of the above

Answer [=]

B

Explanation:
NoSuchMethodException is a checked exception in Java. Here, you are telling the compiler that the MAIN method throws some exception. So, the program compiles and outputs. As the code is throwing a newly created exception, the execution stops here.

4) What are the difference between Checked and Unchecked exceptions in Java?

A) Unchecked exceptions produce less number of compiler errors than the code with Checked exceptions.

B) Most of the unchecked exceptions have RuntimeException as the superclass. Checked exceptions have different classes are superclasses.

C) Unchecked exceptions cause more number of runtime exceptions at the time of execution. As the Checked exceptions are handled well with TRY CATCH blocks, you will get less number of exceptions or errors at the time of execution.

D) All the above

Answer [=]

D

5) Can you catch an object of type Error in Java using TRY-CATCH blocks?

A) Yes

B) No

C) -

D) -

Answer [=]

A

Explanation:
Yes. You can catch. But do not catch as these indicate some serious problems.

6) Which is the superclass of Error class in Java?

A) RuntimeException

B) Exception

C) Throwable

D) None of the above

Answer [=]

C

Explanation:
Yes. Throwable is the superclass of both Error and Exception classes in Java.

7) An Error is treated like a _____ type of exception by the compiler.

A) Unchecked

B) Checked

C) -

D) -

Answer [=]

A

Explanation:
An Error is treated like an Unchecked exception in Java. So, the compiler does not show any errors for not-handling the errors properly.

8) Which is the exception handling block that closes the used resources inside of it automatically in Java?

A) TRY

B) CATCH

C) FINALLY

D) TRY with resource

Answer [=]

D

Explanation:
Yes. TRY-with-resource block automatically closes the specified resource automatically.

9) Choose the correct syntax of a try-with-resource in Java.

A)

```
try(var1 initialized; var2 initialized;)
{

}
```

B)

```
try(var1; var2;)
{

}
```

C)

```
try(var1 initialized, var2 initialized;)
{
```

```
}
```

D)

```
try(var1 uninitialized, var2 uninitialized;)
{

}
```

Answer [=]

A

Explanation:
You should always initialize variables used inside TRY-WITH-RESOURCE block in order to close those automatically.

10) Who closes the resources using code in a TRY-with-resources block in Java?

A) Compiler

B) Runtime

C) Programmer

D) None

Answer [=]

A

Explanation:
The compiler inserts the necessary code inside the FINALLY block (whether explicitly written or not by the programmer) to close the resources.

11) What is the output of the below Java program with Try With Resource block?

```
import java.io.*;

public class ExceptionTest12
{
  public static void main(String[] args)
  {
    try(BufferedWriter bw = new BufferedWriter(new FileWriter("abc.txt"));)
    {
      System.out.println("Inside Try With Resource block.");
    }
```

```
    catch(Exception e)
    {
    }
  }
}
```

A) Inside Try With Resource block.

B) No output

C) Compiler error

D) None of the above

Answer [=]

A

Explanation:
The resource variable "bw" referring BufferedWriter will be closed automatically. Otherwise, the memory will be filled and wasted.

12) Is it possible to write Java code efficiently without using TRY-WITH-RESOURCE blocks?

A) Yes

B) No

C) -

D) -

Answer [=]

A

Explanation:
Yes. Use the normal TRY, CATCH and FINALLY blocks. Keep the code that closes resources inside FINALLY block.

13) A try-with-resource supports only the objects of classes to close automatically that implement ___ Interface in Java.

A) java.lang.Closeable

B) java.lang.AutoCloseable

C) Both A and B

D) None

Answer [=]

C

Explanation:
Yes. You can use the objects of classes that implement java.lang.Closeable or java.lang.AutoCloseable inside TRY-WITH-RESOURCE.

14) What is the output of the below Java program with nested exceptions?

```
public class ExceptionTest14
{
  public static void main(String[] args)
  {
    try
    {
      int a=10/1;
      try
      {int b=20/1;}
      catch(Exception e1)
      { System.out.println("b=20"); }
    }
    catch(Exception e2)
    {System.out.println("a=10");}
  }
}
```

A)

```
a=10
b=20
```

B)

```
b=20
a=10
```

C) Compiler error

D) No output

Answer [=]

C

Explanation:
As no exception is thrown, no output is produced.

15) Choose the right statement about nesting of try-catch-finally in Java?

A)

```
try
{
  try{}
  catch(Exception e1){}
  finally{}
}
catch(Exception e2)
{}
```

B)

```
try
{
  try{}
  catch(Exception e1){}
  finally{}
}
catch(Exception e2)
{
  try{}
  catch(Exception e3){}
  finally{}
}
```

C)

```
try
{
}
catch(Exception e2)
{
}
finally
{
  try{}
  catch(Exception e3){}
  finally{}
}
```

D) All the above

Answer [=]

D

Explanation:
Nesting of exception handling blocks is allowed in any order. So, a try-catch-finally can be kept inside another TRY or CATCH or FINALLY block without errors.

16) The variables initialized inside a TRY-with-resource are treated like ____ variables in Java.

A) static

B) instance

C) final

D) None of the above

Answer [=]

C

Explanation:
A try-with-resource statement treats the variables like FINAL. So, reinitializing is not possible and the compiler throws error.

17) An exception of user-generated-type is treated like a ___ exception.

A) Checked

B) Unchecked

C) -

D) -

Answer [=]

A

Explanation:
You can create a custom exception class simply by subclassing the class Exception. You can throw or catch this user-generated exception like any predefined exception.

18) What is the output of the Java code with custom exceptions?

```
public class ExceptionTest15
{
  void show(int a) throws MyException
  {
    System.out.println("Hello Custom Exception");
    int b = a/0;
  }
```

```
  public static void main(String[] args)
  {
    ExceptionTest15 obj = new ExceptionTest15();
    obj.show(5);
    System.out.println("Bye Custom Exception");
  }
}

class MyException extends Exception
{
  MyException(){ super();}
  MyException(String name){ super(name); }
}
```

A)

```
Hello Custom Exception
Bye Custom Exception
```

B)

```
Hello Custom Exception
```

C)

```
Bye Custom Exception
```

D) Compiler error

Answer [=]

D

Explanation:
You need to catch or throw the custom exception as it is treated like checked. So, you can not simply call the method throwing a checked exception. If it is non-checked, the compilation completes fine.

1) What is an Exception in Java?

A) An exception is like a generic error error occurring during the time of execution of program

B) An exception may occur due to memory or hardware issues

C) An exception leads to bad experience for an End user of the software

D) All the above

Answer [=]

D

2) All Java exceptions can be handled gracefully. State TRUE or FALSE.

A) TRUE

B) FALSE

C) -

D) -

Answer [=]

A

Explanation:
Yes. Simply surround all the code within TRY and CATCH.

3) Are the classes Error and Exception similar in Java?

A) No

B) YES

C) -

D) -

Answer [=]

A

Explanation:
No. An error can not be handled smooth without breaking the flow of execution.

4) Which is the super class in Java that bundles all classes to deal with exceptions and errors?

A) Error

B) Exception

C) Throwable

D) Throw

Answer [=]

C

5) Which are the keywords used to handle exceptions in Java?

A) TRY, CATCH

B) FINALLY

C) THROW, THROWS

D) All the above

Answer [=]

D

Explanation:
The 5 exception handling keywords are try, catch, finally, throw and throws.

6) Where should you keep your Java code for checking against runtime exceptions?

A) Inside TRY block

B) Inside CATCH block

C) inside FINALLY block

D) All the above

Answer [=]

A

Explanation:

```
try{
  //Java code with expected exceptions
}
```


7) Where should you keep your Java code that helps to take action on the exceptions raised inside TRY block?

A) One more TRY block

B) CATCH block

C) FINALLY block

D) None of the above

Answer [=]

B

Explanation:
CATCH block follows TRY block. Based on the type of Exception raised or thrown, you can write some fall back code inside CATCH block.

8) What is the output of the below Java code with Exceptions?

```
public class ExceptionTest1
{
  public static void main(String[] args)
  {
    try
    {
      int a=9, b=0;
      int c = a/b;
      System.out.println("Exception occurred.");
    }
    catch(Exception e)
    {
      System.out.println("Catching an Exception.");
    }
  }
}
```

A)

```
Exception occurred.
Catching an Exception.
```

B)

```
Exception occurred.
```

C)

```
Catching an Exception.
```

D) No output

Answer [=]

C

Explanation:
The first PRINTF with "Exception occurred." is not printed as the program flow already broke because of exception caused by division by zero.

9) If an exception occurs, all the lines of Java code that is below it stops executing. State TRUE or FALSE.

A) TRUE

B) FALSE

C) -

D) -

Answer [=]

A

10) Java exceptions are handled by ____.

A) Java Compiler

B) Java Runtime

C) -

D) -

Answer [=]

B

Explanation:
Yes. Exceptions may or may not occur. It is the Java Runtime that monitors and creates an Exception object that will be handed over to the suitable CATCH block.

11) Can you add more than one CATCH block when handling exceptions in Java?

A) YES

B) NO

C) -

D) -

Answer [=]

A

Explanation:
Yes. You can add as many CATCH blocks as you want with each Block catching one unique Exception type object.

12) A TRY block is the reason for generating a manual exception or an unexpected Runtime exception. State TRUE or FALSE.

A) TRUE

B) FALSE

C) -

D) -

Answer [=]

A

Explanation:
In practice, exceptions are unexpected bugs which spoil the user experience.

13) What is the output of the below Java code with Exceptions?

```
public class ExceptionTest2
{
  public static void main(String[] args)
  {
    try
    {
      int ary[] = {10, 20, 30};
      int tempt = ary[4];
    }
    catch(ArrayIndexOutOfBoundsException e1)
    {
      System.out.println(e1.getMessage());
    }
    catch(Exception e2)
    {
      System.out.println("Some exception");
    }
  }
}
```

A)

```
Index 4 out of bounds for length 3
```

B)

```
Index 4 out of bounds for length
```

```
Some exception
```

C)

```
Some exception
```

D) No exception occurs

Answer [=]

A

Explanation:
IndexOutOfBoundsException is raised by TRY block. Observe the order of catching the exceptions. Always catch a Subclass exception before a Superclass exception.

14) Choose the correct way of writing multiple CATCH blocks in Java.

A)

```
try {
  int num = 10/0;
}
catch(Exception e1)
{
  System.out.println("EXCEPTION");
}
catch(ArithmeticException e2)
{
  System.out.println("ARITHMETIC EXCEPTION");
}
```

B)

```
try {
  int num = 10/0;
}
catch(ArithmeticException e2)
{
  System.out.println("ARITHMETIC EXCEPTION");
}
catch(Exception e1)
{
  System.out.println("EXCEPTION");
}
```

C) -

D) -

Answer [=]

B

Explanation:
An exception of Subclass type should be caught before an exception of Superclass type. The ArithmeticException is a subclass of superclass Exception. So, you should first catch the ArithmeticException. Otherwise, all the next CATCH blocks will become unreachable and compiler throws error.

15) Where do you keep the Java code that should be executed at all times even if an exception occurs?

A) inside TRY block

B) inside CATCH block

C) inside FINALLY block

D) All the above

Answer [=]

C

Explanation:
A FINALLY block in Java contains the code that will be executed whether an exception occurs or not. It usually contains the code to close the resources like files and devices. So, this FINALLY block is fail-safe.

16) What is the output of the below java code with exception handling blocks?

```
public class ExceptionTest4
{
  public static void main(String[] args)
  {
    try
    {
    }
    catch(Exception e)
    {
    }
    finally
    {
      System.out.println("FINALLY block executed");
    }
```

```
  }
}
```

A) No output

B) FINALLY block executed

C) Compiler error

D) None

Answer [=]

B

Explanation:
Even the TRY block contains no code to execute, the finally block will be called.

17) What is the output of the below Java code with exception handling blocks?

```
public class ExceptionTest5
{
  public static void main(String[] args)
  {
    int ary[] = new int[2];
    ary[10] = 5;
    try
    {
      int number= 2/0;
    }
    catch(Exception e)
    {
      System.out.println("Divide by Zero");
    }
    finally
    {
      System.out.println("Inside FINALLY block");
    }
  }
}
```

A) Inside FINALLY block

B)

```
Divide by Zero
Inside FINALLY block
```

C) Compiler error

D) No output

Answer [=]

C

Explanation:
As the exception (ArrayIndexOutOfBoundsException) occurs even before the TRY block, program execution halts and all the remaining code including FINALLY block will not be executed. Even if a single line of TRY block code is executed, its corresponding FINALLY block will be executed.

18) What is the output of the Java code with FINALLY block and RETURN statement?

```
public class ExceptionTest6
{
  static void show()
  {
    try
    {
    System.out.println("inside TRY");
    return;
    }
    finally
    {
    System.out.println("inside FINALLY");
    }
  }
  public static void main(String[] args)
  {
    show();
  }
}
```


A)

```
inside TRY
```

B)

```
inside TRY
inside FINALLY
```

C)

```
inside FINALLY
```

D) Compiler error

Answer [=]

B

Explanation:
Even if a RETURN statement is present at the last line of TRY block, the control is not returned to the calling method. The JVM searches for the suitable FINALLY block and executes it before returning. So, the FINALLY block has higher priority than a RETURN statement.

19) Choose the correct way of writing TRY CATCH FINALLY blocks in Java.

A)

```
try{}
catch{}
finally{}
```

B)

```
try{}
finally{}
```

C)

```
try{}
catch{}
```

D) All the above

Answer [=]

D

20) The FINALLY block should always be accompanied by a ___ block in Java.

A) TRY

B) CATCH

C) FINALLY

D) None

Answer [=]

A

Explanation:
Yes. To write a FINALLY block, there should be a TRY block always.

21) The CATCH block should always be accompanied by a ____ block in Java.

A) TRY

B) CATCH

C) FINALLY

D) None

Answer [=]

A

Explanation:
Yes. There is no value to the CATCH block without a TRY block. The Java compiler throws error if you write CATCH blocks without a TRY.

22) How many extra TRY blocks may be present with an existing TRY block in Java?

A) 0

B) 1

C) 2

D) 3

Answer [=]

A

Explanation:
There can be only one TRY block present in try-catch-finally series of blocks.

23) How many CATCH blocks may be present in try-catch-finally series of blocks in Java?

A) 1

B) 2

C) 3

D) any number of catch blocks

Answer [=]

D

Explanation:
Yes. You can write as many CATCH blocks as you want without duplicating.

24) What is an Unchecked Exception in Java?

A) An exception that need not be handled

B) An exception that may be declared without compiler error

C) Exceptions which will ignored by the compiler for not catching or for throwing

D) All the above

Answer [=]

D

Explanation:
Yes. Unchecked Java exceptions will not be checked by the compiler for mishandling or throwing.

25) What is a Checked Exception in Java?

A) An exception which should be thrown inside the method along with declaring that the method throws it.

B) An exception which should be handled using TRY CATCH statements.

C) Exceptions for which the compiler throws error without completing compilation for not handling properly

D) All the above

Answer [=]

D

Explanation:
Yes. Checked exceptions must be handled properly.

26) Choose which are Java unchecked exceptions below?

A) NullPointerException, ArithmeticException

B) IndexOutOfBoundException, ArrayIndexOutOfBoundsException

C) IllegalArgumentException, StringIndexOutOfBoundsException

D) All the above

Answer [=]

D

Explanation:
These unchecked exceptions are subclasses of RuntimeException class.

27) Choose which are Java checked exceptions below?

A) ClassNotFoundException

B) NoSuchMethodException, NoSuchFieldException

C) InterrruptedException, IllegalAccessException

D) All the above

Answer [=]

D

Explanation:
These checked exceptions must always accompany with TRY-CATCH blocks and THROW-THROWS statements.

1) What are the features of an Object Oriented Programming (OOPs)?

A) Inheritance

B) Encapsulation

C) Polymorphism

D) All the above

Answer [=]

D

2) What are the features reused using Inheritance in Java?

A) Methods

B) Variables

C) Constants

D) All the above

Answer [=]

D

Explanation:
Variables and Methods are reused through inheritance. Constants are nothing but variables only if they hold some value.

3) The class that is being inherited or subclassed is called ___.

A) Subclass

B) Superclass

C) -

D) -

Answer [=]

B

Explanation:
Superclass or Super-Class

4) The class that inherits an already defined class is called ___.

A) Subclass

B) Superclass

C) -

D) -

Answer [=]

A

Explanation:
Subclass

5) Java language supports ___ type of inheritance.

A) Multiple Inheritance

B) Multi-Level Inheritance

C) -

D) -

Answer [=]

B

Explanation:
Multi-Level Inheritance is somewhat complicated.

6) You should use Inheritance when there is an IS-A relationship between classes. State TRUE or FALSE.

A) TRUE

B) FALSE

C) -

D) -

Answer [=]

A

Explanation:
Yes. One simple example is ANIMAL Superclass and HORSE Subclass. HORSE is-a/is-an ANIMAL.

7) What are the types of Inheritances (Whether Java supports or not) available in Object-Oriented Programming Languages?

A) Single Inheritance

B) Multi-Level Inheritance, Hierarchical Inheritance

C) Multiple Inheritance, Hybrid Inheritance

D) All the above

Answer [=]

D

Explanation:
Java supports extending from only one Superclass. Multilevel inheritance is completely supported by Java. Whereas Multiple and Hybrid inheritances are based on Multiple-Superlclasses scenario and hence not supported by Java.

8) In a Single inheritance, Class B inherits only from Class A. State TRUE or FALSE.

A) TRUE

B) FALSE

C) -

D) -

Answer [=]

A

Explanation:
True.

9) In a Multi Level Inheritance Class-C inherits from Class-B and Class-B inherits from Class-A. State TRUE or FALSE.

A) TRUE

B) FALSE

C) -

D) -

Answer [=]

A

Explanation:
True

10) In a Multi-Level Inheritance in Java, the last subclass inherits methods and properties of ____.

A) Only one immediate Superclass

B) Few classes above it.

C) All classes above it

D) None

Answer [=]

C

Explanation:
Yes. The last class inherits all of the properties and methods of all of the classes above it in the chain.

11) When a Class inherits two superclasses (not in Java), it is called ____ inheritance.

A) Multilevel inheritance

B) Single Inheritance

C) Multiple Inheritance

D) None

Answer [=]

C

Explanation:
Multiple Inheritance

12) A Subclass can become a Superclass to another class extending from it in Java. State TRUE or FALSE.

A) TRUE

B) FALSE

C) -

D) -

Answer [=]

A

Explanation:
Yes, true.

13) You can not inherit a Superclass'es constructor even after using inheritance in Java. State TRUE or FALSE.

A) TRUE

B) FALSE

C) -

D) -

Answer [=]

A

Explanation:
Yes. True. A Constructor is class-specific. It is not like a method that can be shared.

14) Find Superclass and Subclass in the below Java code snippet?

```
class B
{
  void show(){}
}
class A
{
  void hide(){}
}
```

A) B is superclass and A is subclass.

B) A is superclass and B is a subclass.

C) There is no superclass or subclass present.

D) None

Answer [=]

C

Explanation:
As there is no use of the "extends" keyword, inheritance does not come into the picture. So, there is no superclass or subclass present in the above example.

15) Find Superclass and Subclass in the below Java program?

```
class Liquid
{
  void pour(){}
}
class Juice extends Liquid
{
  void filter(){}
}
```

A) The Liquid is a superclass and Juice is a subclass.

B) The Liquid is a Subclass and Juice is a Superclass.

C) There is no superclass or subclass

D) None

Answer [=]

A

Explanation:
As the Juice class is extending from the Liquid class, Juice is a subclass and Liquid is a superclass. Simply put, the class before the "extends" keyword is always a Subclass.

16) Which is the keyword used to implement inheritance in Java?

A) extends

B) implements

C) instanceof

D) None

Answer [=]

A

Explanation:
The keyword "extends" tells the compiler that the class on the left side is subclassing the class on the right side.

17) What is the output of the below Java program with inheritance?

```
class Sweet
{
  void price()
  {
    System.out.print("Sweet=$10 ");
  }
}
class Sugar extends Sweet
{
  void price()
  {
    super.price();
    System.out.print("Sugar=$20");
  }
}
public class JavaInheritance1
{
  public static void main(String[] args)
  {
    Sugar su = new Sugar();
    su.price();
  }
}
```


A) Sweet=$10 Sugar=$20

B) Sweet=$10 Sugar=$10

C) Sweet=$20 Sugar=$20

D) Compiler error

Answer [=]

A

Explanation:
Notice the use of the keyword "super". Using this keyword, you can call superclass's methods and variables.

18) Can you call it a full-fledged inheritance of using ABSTRACT classes and INTERFACES in Java?

A) NO

B) YES

C) -

D) -

Answer [=]

A

Explanation:
No. Abstract classes and Interfaces do not define a class completely. So, it is not called a full-fledged inheritance of using those.

19) To control inheritance to different classes and levels, Java provides ____.

A) Return types like the void, int, float, double and other object types

B) Static keyword

C) Access modifiers like default, public, protected, private

D) None

Answer [=]

C

Explanation:
Access modifiers like default (not a keyword), public, protected and private changed the visibility of a method, variable or a class. Even the keyword "package" also allows grouping of classes and control inheritance levels to some extent.

20) To stop or block inheriting a given class, the ___ keyword is used before the class.

A) static

B) private

C) final

D) none of the above

Answer [=]

C

Explanation:
You can not subclass a class that is marked FINAL.

```
final class CLASS_NAME //Can not subclass
{

}
class SUBCLASS extends CLASS_NAME //ERROR
{

}
```

1) What is the maximum number of levels possible in a Multilevel Inheritance in Java?

A) 8

B) 16

C) 32

D) No maximum level

Answer [=]

D

Explanation:
There is no limit to the number of levels in a multilevel inheritance chain in Java.

2) What is the output of the below java program with Constructors and Inheritance?

```
class Processor
{
  Processor()
  {
    System.out.print("Inside Processor() Constructor. ");
  }
}
class I3Processor extends Processor
{
  I3Processor()
  {
    System.out.print("Inside I3Processor() Constructor. ");
  }
}
class I5Processor extends I3Processor
{
  I5Processor()
  {
    System.out.print("Inside I5Processor() Constructor. ");
  }
}

public class JavaInheritance2
{
  public static void main(String[] args)
  {
    I5Processor i5 = new I5Processor();
  }
}
```

A) Inside I5Processor() Constructor. Inside I3Processor() Constructor. Inside Processor() Constructor.

B) Inside I5Processor() Constructor. Inside I5Processor() Constructor. Inside I5Processor() Constructor.

C) Inside Processor() Constructor. Inside I3Processor() Constructor. Inside I5Processor() Constructor.

D) Compiler error

Answer [=]

C

Explanation:
The example demonstrates the Invoking of constructors using a Multilevel Inheritance. Always, the default superclass's constructor is invoked. And then, the subclass's constructor is invoked.

```
Processor <-- I3Processor <-- I5Processor
```

3) What is the output of the below Java program with Constructors using Inheritance?

```
class Ant
{
  Ant(String name)
  {
    System.out.print("Inside Ant(String) Constructor. ");
  }
}
class WildAnt extends Ant
{
  WildAnt()
  {
    System.out.print("Inside WildAnt() Constructor. ");
  }
}

public class Inheritance3
{
  public static void main(String[] args)
  {
    WildAnt wa = new WildAnt();
  }
}
```

A) Inside WildAnt() Constructor.

B) Inside Ant(String) Constructor. Inside WildAnt() Constructor.

C) Inside WildAnt() Constructor. Inside WildAnt() Constructor.

D) Compiler error

Answer [=]

D

Explanation:
Compiler throws an error "implicit constructor of Ant class is undefined". Once you define a Constructor with some arguments, the compiler does not add a default constructor with zero arguments. To subclass a class, defining a default constructor is mandatory.

4) If a class is not subclassed by any class, then defining a default constructor is not mandatory in Java. State TRUE or FALSE.

A) TRUE

B) FALSE

C) -

D) -

Answer [=]

A

Explanation:
True.

5) What is the output of the below Java program?

```
final class Bus
{
  void show()
  {
    System.out.print("Generic Bus. ");
  }
}
class ElectricBus extends Bus
{
  void show()
  {
    System.out.println("Electric Bus. ");
  }
}
public class Inheritance4
```

```
{
  public static void main(String[] args)
  {
    ElectricBus eb = new ElectricBus();
    eb.show();
  }
}
```

A) Generic Bus

B) Electric Bus

C) Generic Bus. Electric Bus.

D) Compiler error.

Answer [=]

D

Explanation:
Notice the use of the keyword "final" before the superclass BUS. You can not subclass a class that is marked final.

6) A Superclass reference can refer to a Subclass Object without casting. State TRUE or FALSE.

A) TRUE

B) FALSE

C) -

D) -

Answer [=]

A

Explanation:
Yes.

7) A superclass reference can not be used to invoke a method or variable of the subclass. State TRUE or FALSE.

A) TRUE

B) FALSE

C) -

D) -

Answer [=]

A

Explanation:
Yes. A superclass reference knows only about the methods and properties of the same class but not the subclass.

8) A subclass object can be used to invoke (call) a method or property of the superclass. State TRUE or FALSE.

A) TRUE

B) FALSE

C) -

D) -

Answer [=]

A

Explanation:
Yes, True.

9) What is the output of the below Java program on the references of Superclass and Subclass?

```
class Food
{
  void show()
  {
    System.out.print("FOOD ");
  }
}
class Bread extends Food
{
  void toast()
  {
    System.out.print("TOASTED ");
  }
}
public class Inheritance5
{
  public static void main(String[] args)
  {
    Food foo = new Food();
```

```
    foo.show();
    Food foo2 = new Bread();
    foo2.show();
    Bread br = new Bread();
    br.toast();
    br.show();
  }
}
```

A) FOOD FOOD FOOD FOOD

B) FOOD FOOD TOASTED FOOD

C) FOOD TOASTED FOOD FOOD

D) Compiler error

Answer [=]

B

Explanation:
You can only invoke methods of the Superclass using a Superclass reference variable pointing to a Subclass object.

10) What is the output of the below Java program using Inheritance?

```
class Furniture
{
  void show()
  {
    System.out.println("Made of Wood. ");
  }
}
class Sofa extends Furniture
{
  void addCushion()
  {
    System.out.println("Added. ");
  }
}
public class Inheritance6
{
  public static void main(String[] args)
  {
    Furniture fur = new Sofa();
    fur.addCushion();
  }
}
```

A) Added.

B) No output

C) Added. Made of Wood.

D) Compiler error

Answer [=]

D

Explanation:
Yes. The compiler throws an error saying "The method addCushion() is undefined for the type Furniture". It means that you can not call a method of subclass using a superclass reference even though it is pointing to a subclass object.

1) An interface in Java is like a 100% ____.

A) abstract class

B) public class

C) inner class

D) anonymous class

Answer [=]

A

Explanation:
Yes. If 100% of methods in an abstract class are marked abstract, then it is comparable to an interface in Java.

2) A Java Interface is not considered a class. State TRUE or FALSE.

A) TRUE

B) FALSE

C) -

D) -

Answer [=]

A

Explanation:
True. A class and an Interface have different inheritance rules.

3) Choose the correct syntax below for defining an Interface in Java.

A)

```
interface NAME
{
  //abstract methods
}
```

B)

```
abstract interface NAME
{
  //abstract methods
}
```

C)

```
public interface NAME
{
  //abstract methods
}
```

D) All the above

Answer [=]

D

Explanation:
There is no need to explicitly mention ABSTRACT keyword to define an interface.

4) Choose a correct statement about Java Interfaces?

A) Interface contains only abstract methods by default.

B) A Java class can implement multiple interfaces

C) An Interface can extend or inherit another Interface.

D) All the above

Answer [=]

D

5) A Java Class inherits Constants and Methods of an Interface using ____ keyword.

A) INTERFACE

B) IMPLEMENTS

C) EXTENDS

D) All the above

Answer [=]

B

Explanation:
IMPLEMENTS keyword

6) What is the output of the below Java program with an Interface?

```
interface Bus
{
  void move();
```

```
}
class ElectricBus implements Bus
{
  public void move()
  {
    System.out.println("Implemented  move() method.");
  }
}
public class InterfaceTest1
{
  public static void main(String[] args)
  {
    new ElectricBus().move();
  }
}
```

A) No output

B) Implemented move() method.

C) Compiler error

D) None of the above

Answer [=]

B

7) What is the output of the below Java program with an Interface?

```
interface Car
{
  int basePrice=1000;
}
public class InterfaceTest2 implements Car
{
  void changePrice()
  {
    basePrice = 2000;
    System.out.print(basePrice);
  }
  public static void main(String[] args)
  {
    new InterfaceTest2().changePrice();
  }
}
```

A) 1000

B) 2000

C) Compiler error

D) None of the above

Answer [=]

C

Explanation:
Java Interface treats its variables like constants. So, the classes implementing Interfaces, can not reassign values to the variables.

8) What is the output of the below Java program with an Interface?

```
interface Book
{
  char type='C';
}
public class InterfaceTest3
{
  public static void main(String[] args)
  {
    System.out.println(new Book().type);
  }
}
```

A) C

B) No output

C) Compiler error

D) None of the above

Answer [=]

C

Explanation:
You can not instantiate an Interface in Java. So, using the keyword "new" does not create new objects of an Interface.

9) All Interface variables are ___ by default in Java.

A) public

B) final

C) public and final

D) None

Answer [=]

C

Explanation:
Yes, public and final. In other words, these are constants.

10) All Interface methods in Java are ____ by default.

A) public

B) abstract

C) public and abstract

D) None of the above

Answer [=]

C

Explanation:
Interface automatically marks all its methods as public and abstract. So, you need not add these keywords again while writing the program.

11) A Class implementing an Interface can use ____ access modifier before the implemented methods.

A) private

B) protected

C) public

D) All the above

Answer [=]

C

Explanation:
Only a "public" access modifier is allowed.

12) A Java Class implementing an Interface can define a variable with the same name as that of the Interface constant. State TRUE or FALSE.

A) TRUE

B) FALSE

C) -

D) -

Answer [=]

A

Explanation:
True.

13) What is the output of the below Java program with an Interface?

```
interface Worm
{
  int teeth=2;
}
class BookWorm implements Worm
{
  int teeth=4;
  void show()
  {
    teeth= 5;
    System.out.println("Teeth: " + teeth);
  }
}
public class InterfaceTest4
{
  public static void main(String[] args)
  {
    new BookWorm().show();
  }
}
```


A) Teeth: 4

B) Teeth: 5

C) Compiler error as teeth is a constant in Worm interface.

D) None of the above

Answer [=]

B

Explanation:
You can reassign an interface's constant. You can define a variable with the same name in the implementing class.

14) A Java Interface can not declare constructors. State TRUE or FALSE.

A) TRUE

B) FALSE

C) -

D) -

Answer [=]

A

Explanation:
True. You can define a constructor inside an Interface.

15) What is the output of the below Java program with an Interface?

```
interface Floor
{
  Floor(){ }
}
public class InterfaceTest5
{
  public static void main(String[] args)
  {
    System.out.print("Floor");
  }
}
```

A) Floor

B) No output

C) Compiler error

D) None

Answer [=]

C

Explanation:
Interfaces can not have constructors.

16) Java 8 (Java 1.8) introduced the ___ feature.

A) Default methods

B) Static methods

C) Default and Static methods

D) None of the above

Answer [=]

C

Explanation:
Yes, static and default methods. Remember that "default" is a keyword.

17) Java Interface static methods have ___ compatibility with the existing project code.

A) Forward

B) Backward

C) Both Forward and Backward

D) -

Answer [=]

C

Explanation:
Forward compatibility means, the implementing classes may be modified to access these static methods. It is optional. It does not throw exceptions. The existing classes still do not utilize these new static methods of the interface. So, it is backward compatible too.

18) Java Interface DEFAULT methods have ___ compatibility with the existing project code.

A) Forward

B) Backward

C) Backward and Forward

D) -

Answer [=]

C

Explanation:
Backward and Forward Compatibility. It means, the existing project-code compiles as it is without asking for overriding the newly added Default method inside the Interface. Without the keyword DEFAULT, the project build fails. All the new Classes start implementing these default methods. It is forward compatibility.

19) The DEFAULT methods of an Interface are suitable mostly for ___ type of projects.

A) Open Source (Public Repositories)

B) Closed Source (Private Repositories)

C) -

D) -

Answer [=]

A

Explanation:
Open Source projects do not know how many organizations or users have been dependent on the project. So, it is advised to take advantage of the DEFAULT methods of an Interface to introduce new features.

20) Is it possible to remove the keyword DEFAULT and make the method abstract again in an Interface, if the Interface belongs to a Closed-Source project?

A) Yes

B) No

C) -

D) -

Answer [=]

A

Explanation:
Yes. Closed source projects can still introduce new features using the same keyword DEFAULT. Once they complete the implementation of all the DEFAULT methods in the implementing classes, they can completely remove default methods and provide only abstract methods. The end-user of a Closed-Source project is the company itself that developed it.

21) The annotation used in Java to override the method of a super-class or interface by the subclass or implementing class is ___.

A) @override

B) @Override

C) @super

D) @subclass

Answer [=]

B

Explanation:
@Override

22) It is ___ to override the static method of an Interface in Java.

A) possible

B) not possible

C) -

D) -

Answer [=]

B

Explanation:
Not possible to override the static method. The compiler shows an error.

23) A Java static method can not be ___.

A) private or protected

B) final

C) abstract

D) All the above

Answer [=]

D

Explanation:
You can not use the keywords, private, protected, final and abstract, before a static method of an Interface.

24) Which is the missing java code in the class implementing an Interface below?

```
interface Linein
{  void addInput();     }

interface Lineout
{  void addOutput(); }

class Speaker implements Linein, Lineout
{
```

```
  //MISSING CODE
}
```

A)

```
class Speaker implements Linein, Lineout
{
  @Override
  public void addOutput() { }

  @Override
  public void addInput() {  }
}
```

B)

```
class Speaker implements Linein, Lineout
{
  @Override
  public void addOutput() { }
}
```

C)

```
class Speaker implements Linein, Lineout
{
  @Override
  public void addInput() {  }
}
```

D) All the above

Answer [=]

A

Explanation:
As the Speaker class is implementing two interfaces Linein and Lineout, the abstract methods of all the interfaces have to be implemented by the first concrete class.

25) Which is the missing code to successfully compile the below Java program with abstract classes and Interfaces?

```
interface A
{  void a(); }
```

```
abstract class B implements A
{  abstract void b(); }

class C extends B
{
  //Missing methods
}
```

A)

```
@Override
public void a() {  }

@Override
void b() {}
```

B)

```
@Override
public void a() {  }
```

C)

```
@Override
void b() {}
```

D) All the above

Answer [=]

A

Explanation:
The first concrete class should implement all the abstract methods of superclasses and interfaces.

26) A Superinterface is comparable to a Superclass. State TRUE or FALSE.

A) TRUE

B) FALSE

C) -

D) -

Answer [=]

A

Explanation:
True.

27) A Static method of an Interface should be accessed with _____ and a DOT operator.

A) Class Name

B) Interface Name

C) An object of a concrete class

D) None of the above

Answer [=]

B

Explanation:
You should not use object references to access the static method of an Interface. Just use Interface name and DOT (.) operator directly.

1) What is method overriding in Java?

A) Writing a method in a subclass with the same name of superclass's method

B) Mentioning the same return type of the method of the superclass

C) The argument list in the method of subclass and the method of superclass should be the same

D) All the above

Answer [=]

D

2) Method Overriding is useful to add extra functionality or code to a method of subclass with the same name as the inherited method. State TRUE or FALSE.

A) TRUE

B) FALSE

C) -

D) -

Answer [=]

A

3) It is not mandatory to override all or a few methods of the Superclass. State TRUE or FALSE.

A) TRUE

B) FALSE

C) -

D) -

Answer [=]

A

Explanation:
Yes. It is not mandatory. If the inheriting method serves the purpose, use it directly. Otherwise, write your custom code in the overridden method.

4) Why should a method be overridden in Java instead of writing a method with a different name?

A) Large projects heavily depend on inheritance

B) The code-base refers to the same method with the same method signature in different classes of the project

C) It is not possible to change the method calling code at all occurrences of the project. It may break the whole project.

D) All the above.

Answer [=]

D

Explanation:
All the above.

5) The Method-Overloading and Method-Overriding are not the same. State TRUE or FALSE.

A) TRUE

B) FALSE

C) -

D) -

Answer [=]

A

Explanation:
TRUE

6) What is the output of the below Java program with Method Overriding?

```
class Bus
{
  void seatingCapacity()
  {
    System.out.println("Superclass Seats=32");
  }
}
class ElectricBus extends Bus
{
  void seatingCapacity()
  {
    System.out.println("Subclass Seats=20");
  }
  void showInfo()
  {
    seatingCapacity();
    this.seatingCapacity();
  }
}
public class MethodOverriding1
```

```
{
  public static void main(String[] args)
  {
    ElectricBus eb = new ElectricBus();
    eb.showInfo();
  }
}
```

A)

```
Subclass Seats=20
Superclass Seats=32
```

B)

```
Superclass Seats=32
Subclass Seats=20
```

C)

```
Superclass Seats=32
Superclass Seats=32
```

D)

```
Subclass Seats=20
Subclass Seats=20
```


Answer [=]

D

Explanation:
Using the keyword "this" calls the local method of the class but not the method of a superclass.

7) What is the output of the below Java program with Method Overriding and SUPER keyword?

```
class Car
{
  void showTransmission()
  {
    System.out.println("Transmission Manual");
```

```
  }
}
class ElectricCar extends Car
{
  void showTransmission()
  {
    System.out.println("Transmission AMT");
  }
  void showInfo()
  {
    this.showTransmission();
    super.showTransmission();
  }
}
public class MethodOverriding2
{
  public static void main(String[] args)
  {
    ElectricCar ec = new ElectricCar();
    ec.showInfo();
  }
}
```

A)

```
Transmission AMT
Transmission Manual
```

B)

```
Transmission Manual
Transmission AMT
```

C)

```
Transmission Manual
Transmission Manual
```

D)

```
Transmission AMT
Transmission AMT
```


Answer [=]

A

Explanation:
The keyword "super" calls the method of a Superclass.

8) Identify INVALID Java Method Overriding in the below code snippets?

Follow the notation "superclassMethod" and "subclassMethod".

A)

```
void superclassMethod(int a, float b){ }

void subclassMethod(int a, float b) { }
```

B)

```
void superclassMethod(){ }

void subclassMethod(){ }
```

C)

```
int superclassMethod(int a, float b){ }

void subclassMethod(int a, float b) { }
```

D) None

Answer [=]

C

Explanation:
The return types are different. So, it is not a successful method override.

9) A successful Method Overriding calls the method of ___ in Java.

A) Superclass

B) Subclass

C) -

D) -

Answer [=]

B

Explanation:
The method overriding is implemented to give preference to the method of a Subclass.

10) A failed method overriding calls the method of a ___ in Java.

A) Superclass

B) Subclass

C) Superclass or Subclass

D) None

Answer [=]

C

Explanation:
If a method override fails, the JVM may call the method of either a Superclass or Subclass. It depends on the parameters passed in the method call.

11) What is the output of the below Java program with method overriding?

```
class Cat
{
  int jumpingHeight(int weight)
  {
    System.out.println(10);
    return 10;
  }
}
class WildCat extends Cat
{
  void jumpingHeight(int weight)
  {
    System.out.println("20");
  }
}
public class MethodOverriding3
{
  public static void main(String[] args)
  {
    WildCat wc = new WildCat();
    wc.jumpingHeight(30);
  }
}
```


A) 10

B) 20

C) 30

D) Compiler error

Answer [=]

D

Explanation:
If the argument list is the same, the return types can not be the incomptible-types. So, the compiler reports an error "The return type is incompatible with Cat.jumpingHeight(int)".

12) What is the output of the below Java program with Method Overriding?

```
class Sparrow{ }
class BigSparrow extends Sparrow { }
class Cat2
{
  Sparrow jumpingHeight(int weight)
  {
    System.out.println(40);
    return new Sparrow();
  }
}
class WildCat2 extends Cat2
{
  BigSparrow jumpingHeight(int weight)
  {
    System.out.println("50");
    return new BigSparrow();
  }
}
public class MethodOverriding4
{
  public static void main(String[] args)
  {
    WildCat2 wc = new WildCat2();
    wc.jumpingHeight(80);
  }
}
```


A) 40

B) 50

C) 80

D) Compiler error

Answer [=]

B

Explanation:

It is perfectly alright to use a subclass type return type when overriding a method in Java. BigSparrow is the subclass of Sparrow. Always, the overriding method will be called.

13) What is the output of the below Java program with method overriding?

```
class Steel
{
  void setGrade(int g)
  {
    System.out.print(",GRADE="+g);
  }
}
class CarbonSteel extends Steel
{
  void setGrade(char grade)
  {
    System.out.print(",Grade="+grade);
  }
}
public class MethodOverriding5
{
  public static void main(String[] args)
  {
    Steel s = new CarbonSteel();
    s.setGrade(5);
    s.setGrade('A');
  }
}
```


A) ,GRADE=5,GRADE=A

B) ,GRADE=5,GRADE=65

C) ,Grade=5,Grade=65

D) Compiler error

Answer [=]

B

Explanation:
As the superclass reference "s" is used, it calls the methods of the superclass only. As the method signatures of the "setGrade" method are different with different argument types, it is not a successful override. It is an overloading of the superclass's method.

14) What is the output of the below Java program with method overriding?

```
class Wood
{
  void setQuality(int q)
  {
    System.out.print(",QUALITY="+q);
  }
}
class PlyWood extends Wood
{
  void setQuality(char qual)
  {
    System.out.print(",quality="+qual);
  }
}
public class MethodOverriding6
{
  public static void main(String[] args)
  {
    PlyWood pw = new PlyWood();
    pw.setQuality(10);
    pw.setQuality('B'); //ASCII of B=66
  }
}
```

A) ,QUALITY=10,quality=B

B) ,QUALITY=10,quality=65

C) ,quality=10,quality=B

D) Compiler error

Answer [=]

A

Explanation:
The method "setQuality" is not overridden successfully as the argument types are different. The subclass type reference can call a method of superclass and subclass.

15) What is the output of the below Java program with method overriding?

```
class Amplifier
{
  void addGain(int a)
  {
```

```
    System.out.println((a + 10)+"dB");
  }
}
class DigitalAmplifier extends Amplifier
{
  void addGain(int a)
  {
    super.addGain(a+5);
  }
}
public class MethodOverriding7
{
  public static void main(String[] args)
  {
    DigitalAmplifier da = new DigitalAmplifier();
    da.addGain(12);
  }
}
```

A) 22dB

B) 27dB

C) 22dB 27dB

D) Compiler error

Answer [=]

B

Explanation:
The subclass DigitalAmplifier successfully overrides the method of the superclass "Amplifier". In the subclass's method, an extra gain of 5 is added and passed to the superclass's method. So, it becomes 27(12+5+10).

16) If the method signature of a Superclass and the method signature of a Subclass are the same, then the subclass method is said to be _____ the superclass's method.

A) Overriding

B) Overloading

C) -

D) -

Answer [=]

A

17) A method of a Superclass can not override the method of the Subclass. State TRUE or FALSE.

A) TRUE

B) FALSE

C) -

D) -

Answer [=]

A

Explanation:
True. Only subclass methods can override the methods of a superclass.

18) Method overriding increases the burden on the JVM in terms of runtime checks and resolution. State TRUE or FALSE.

A) FALSE

B) TRUE

C) -

D) -

Answer [=]

B

Explanation:
Yes. The to be called at runtime is decided at runtime based on successful or failed Overriding.

19) What are the advantages of Method Overriding in Java?

A) A subclass can add extra functionality to the overriding method.

B) A subclass can call both the overridden method and overriding method.

C) It supports polymorphism. A superclass reference can be used to call the common method of all subclasses.

D) All the above

Answer [=]

D

20) An Overridden method is the method of ____ class and the overriding method is the method of ___ class.

A) super, sub

B) sub, super

C) super, super

D) sub, sub

Answer [=]

A

Explanation:
An Overriding method belongs to the Subclass and the Overridden method belongs to the Superclass.

21) To successfully override a superclass method in Java, the access modifier of the method of the subclass can be ___ restrictive.

A) Less

B) More

C) Less or Same

D) None

Answer [=]

C

Explanation:
The access modifier can be less restrictive.

```
protected void show() { }
.
.
public void show() { } //public is less restrictive than private.
```


22) Which is the correct way of overriding a method throwing exceptions in Java?

A)

```
void show() throws IOException{ }
.
.
void show() throws FileNotFoundException{ }
```

B)

```
void show() throws IOException{ }
.
.
void show() throws ArithmeticException{ }
```

C)

```
void show() throws ArithmeticException{ }
.
.
void show() throws IllegalFormatException{ }
```

D) None

Answer [=]

A

Explanation:
The exception thrown by the subclass's method can be a subclass type of the exception thrown by the superclass's method. The superclass of FileNotFoundException is IOException only.

1) What is a package in Java?

A) A Package is a collection of files of type Java Class, Interfaces, or Abstract Class

B) A Package is simply a Directory or Folder with Java Classes

C) A Package usually contains Java Classes written for a specific purpose or problem

D) All the above

Answer [=]

D

2) Choose the correct syntax of a Java Package below.

A)

```
package PACKAGE_NAME;
```

B)

```
package PACKAGE_NAME.*;
```

C)

```
pkg PACKAGE_NAME;
```

D)

```
pkg PACKAGE_NAME.*;
```


Answer [=]

A

Explanation:
A package declaration statement should end with a package name but not with *.

3) The keyword used to declare a Java package is ____.

A) pkg

B) package

C) pkge

D) None of the above

Answer [=]

B

Explanation:
package

4) What is the maximum number of Java Class files that can be kept inside a single Java Package?

A) 8

B) 64

C) 128

D) Unlimited

Answer [=]

D

Explanation:
There is no limit. A Java package can accommodate as many files as the disk can hold.

5) Can you compile a Java file kept inside a directory without mentioning the package name?

```
folder_1
     ----- SomeFile.java
.
.
//SomeFile.java
//no package declaration
//package folder_1; commented
public class SomeFile
{

}
```


A) Yes

B) No

C) -

D) -

Answer [=]

A

Explanation:
Yes. But this Java class can not be imported.

6) The name of a package is the name of the ___ in Java.

A) folder

B) All parent folders separated by DOT symbols

C) All parent packages separated by DOT symbols

D) All the above

Answer [=]

D

7) It is possible to declare a package and import another package within the same Java class file. State TRUE or FALSE.

A) TRUE

B) FALSE

C) -

D) -

Answer [=]

A

Explanation:
Yes. True.

8) The keyword used to import a package into Java class or Interface is ___.

A) import

B) download

C) use

D) None of the above

Answer [=]

A

Explanation:
import

9) Which is the correct syntax to import a Java package below?

A)

```
import PACKAGE1.*;
```

B)

```
import PACKAGE1.CLASS1;
```

C)

```
import PACKAGE1.PACKAGE2.PACKAGE3.*;
```

D) All the above

Answer [=]

D

10) Choose correct declaration and importing of packages in Java.

A)

```
package SOMEPACKAGE;
import PACKAGE_N.*;
```

B)

```
import PACKAGE_N.*;
package SOMEPACKAGE;
```

C)

```
import PACKAGE_M.*;
package SOMEPACKAGE;
import PACKAGE_N.*;
```

D) All the above

Answer [=]

A

Explanation:

There can be only one package statement per class file. The package declaration statement should precede an import statement.

11) You can use the same name for a Parent package and Child package in Java. State TRUE or FALSE.

A) TRUE

B) FALSE

C) -

D) -

Answer [=]

A

12) What is the maximum number of levels or depth up to which sub-packages can be defined in Java?

A) 8

B) 16

C) 32

D) There is no limit

Answer [=]

D

Explanation:
There is no limit.

13) Choose a correct statement below about importing packages into a class.

A) A Java class or interface can import any number of packages.

B) It is advised to import only the required Classes of a package to save memory.

C) Java packages are usually distributed in the form of JAR files.

D) All the above

Answer [=]

D

14) When importing a Package, the Class is actually importing ____.

A) Classes or Interfaces from the package

B) Constants

C) Methods

D) None of the above

Answer [=]

A

15) The package declaration statement should be the first statement in a Java file. State TRUE or FALSE.

A) TRUE

B) FALSE

C) -

D) -

Answer [=]

A

Explanation:
TRUE

16) You can place a comment before the Package Declaration statement in Java. State TRUE or FALSE.

A) TRUE

B) FALSE

C) -

D) -

Answer [=]

A

Explanation:
True

17) How does JAVAC or JAVA (JVM) find the packages that will be used inside classes by an import statement?

A) If the packages are defined on the same root level as the compiling or running class file, Java knows it.

B) You should manually use the CLASSPATH or CP command to include the path of the package or single-class for compiling and running

C) You can copy the JAR files in LIB folder of Java inside Program Files

D) All the above

Answer [=]

D

18) Which is the default Java package that will be auto included (imported) in the classpath while Compiling and Running a Java program?

A) java.io

B) java.util

C) java.net

D) java.lang

Answer [=]

D

Explanation:
java.lang or simply LANG package is imported automatically by the compiler and JVM (Java Virtual Machine)

19) What are the popular Classes or Interfaces inside a Java Language Pack (java.lang)?

A) Byte, Character, Short, Integer, Float, Long, Number

B) Math, String, StringBuffer, StringBuilder

C) Thread, Throwable, Exception, Error

D) All the above

Answer [=]

D

20) Choose a correct way of importing all the classes in the below java program with packages.

```
//Cat.java
package animals;
public class Cat { }

//Dog.java
package animals;
public class Dog { }

//PackageTesting.java
//import statements
```

```
public class
{
  //new Cat();
  //new Dog();
}
```

A)

```
import animals.*;
```

B)

```
import animals.cat;
import animals.Dog;
```

C) Both A and B

D) None

Answer [=]

C

Explanation:
You can import all the classes without mentioning their names with a single STAR. You can also import individual classes using separate import statements.

21) Which is the symbol used to separate a super-package and a sub-package at the time of declaring or importing in a Java program?

A) Dollar ($)

B) Pound (#)

C) Period (.) or DOT

D) Arrow (->)

Answer [=]

C

Explanation:
You can use the notation like package1.package2.package3 to declare a package by name package3 using the keyword "package".

22) Choose a correct statement about using the classes or interfaces or abstract classes inside the packages in our Java program.

A) You can extend the class imported from the package.

B) You can implement the interfaces imported from the package.

C) You can extend the abstract classes imported from the package.

D) All the above

Answer [=]

D

23) Accessing the variables, constants, or methods of a class, imported from a package is subjective to access modifiers like PUBLIC, PRIVATE, PROTECTED and default. State TRUE or FALSE.

A) TRUE

B) FALSE

C) -

D) -

Answer [=]

A

Explanation:
TRUE. As the classes are declared public by default in each .class file, you can import the class successfully. Some variables and methods may be available after subclassing and some are available directly (public variables and methods).

24) What are the uses of a Java Package?

A) A package contains ready-to-use classes written for a specific purpose.

B) Packages are easy to distribute your code. It is nothing but reusability. Instead of writing code afresh, you can take advantage of the existing classes of a package. Simply import it and use.

C) Packages help in maintaining the code easily. Each sub-package may be maintained for more specific purposes. You can reuse the class names in sub-packages or other packages without name clash.

D) All the above

Answer [=]

D

25) What is the output of the below Java program with packages?

```
//MathClas.java
package package1;
public class MathClass
{
  public static int getRandom()
  {
    //returns a random number from 0 to 999
    return ((int) (Math.random() * 1000));
  }
}

//PackageTest3.java
import package1.*;
public class PackageTest3
{
  public static void main(String[] args)
  {
    System.out.print(MathClass.getRandom());
  }
}
```

A) No output

B) Compiler error

C) 484 (some random number)

D) None

Answer [=]

C

Explanation:
You can import all classes of a given package using PACKAGE_NAME.* notation. So, MathClass class will be imported successfully.

1) What is a Type Wrapper or simply a Wrapper in Java?

A) A Wrapper class is an object version of Primitive Data Type

B) A Wrapper provides additional methods for ease of use

C) Wrapper types are useful for using with generic collections

D) All the above

Answer [=]

D

2) Which primitive data types have wrapper classes available?

A) byte, short, int, long

B) float, double

C) boolean, char

D) All the above

Answer [=]

D

3) Choose the correct mapping of Primitive Data Types and Wrapper Classes below?

A) boolean = Boolean, char = Character

B) byte = Byte, short = Short, int = Integer, long = Long

C) float = Float, double = Double

D) All the above

Answer [=]

D

4) State TRUE or FALSE. Java primitive data types work faster compared to their Wrapper counter parts.

A) TRUE

B) FALSE

C) -

D) -

Answer [=]

A

Explanation:

Yes. Developers are encouraged to use primitive types like int, float, long etc for better memory management and performance.

5) Which is the correct way of converting an int value to Integer in below Java code?

A)

```
Integer in = new Integer(9);
```

B)

```
Integer in = Integer.valueOf(9);
```

C)

```

```

```
Integer in = new Integer("9");
```

D) None of the above

Answer [=]

B

Explanation:
The method valueOf() is used to convert a primitive data type like "int" to "Integer".
All constructors of Wrapper classes are deprecated as of Java 9.

6) Which is the Superclass of Wrapper types like Byte, Short, Integer, Long, Float and Double?

A) Math

B) System

C) Number

D) Enum

Answer [=]

C

Explanation:
Number class is the super class of almost all of Wrapper classes.

7) Which are the methods a Wrapper class object can use in Java?

A) byteValue(), shortValue, intValue(), longValue()

B) floatValue(), doubleValue()

C) booleanValue(), charValue()

D) All the above

Answer [=]

D

Explanation:
Except booleanValue() and charValue(), all valueOf() methods are from Number class.

8) Which is the superclass of wrapper classes like Boolean and Character?

A) Number

B) Wrapper

C) Object

D) Math

Answer [=]

C

9) Are the sizes of float and Float same or different in Java?

A) Same

B) Different

C) -

D) -

Answer [=]

A

10) Which are the constant fields available a Wrapper Class object in Java?

A) BYTES, SIZE

B) MAX_VALUE, MAX_EXPONENT

C) MIN_VALUE, MIN_EXPONENT

D) All the above

Answer [=]

D

Explanation:
The above constants are available only to non Boolean wrapper classes.
MIN_EXPONENT and MAX_EXPONENT are available only to Float and Double.

11) What is the output of the below Java code snippet on wrapper classes?

```
System.out.println(Byte.BYTES);
System.out.println(Character.BYTES);
System.out.println(Short.BYTES);
System.out.println(Integer.BYTES);
System.out.println(Long.BYTES);
System.out.println(Float.BYTES);
System.out.println(Double.BYTES);
```

A)

```
1
2
2
4
8
4
8
```

B)

```
1
4
2
4
8
4
8
```

C)

```
1
2
4
8
16
8
16
```

D) None of the above

Answer [=]

A

Explanation:
The sizes of Byte, Character, Short, Integer, Long, Float and Double are 1, 2, 2, 4, 8, 4 and 8 bytes respectively.

12) Choose the correct way of creating a Float wrapper object in Java?

A)

```
Float pi = Float.valueOf(2.54f);
```

B)

```
Float pi = Float.valueOf(2.54);
```

C)

```
Float pi = new Float(2.54f);
```

D)

```
Float pi = new Float("2.54f");
```


Answer [=]

A

Explanation:
If you are using a non string argument, suffix the float number with a "f". Otherwise, the compiler throws error. Passing a string to Float.valueOf("2.54") works fine.

```
Error: The method valueOf(String) in the type Float is not applicable for the arguments (double)
```

13) Choose the correct way of comparing Wrapper objects in Java?

A)

```
float p1 = 10.45f;
float p2 = 5.2f;
if(Float.compare(p1, p2) == 1)
{
  System.out.println("10.45 > 5.2");
}
else
  System.out.println("No Biscuits");

//OUTPUT
10.45 > 5.2
```

B)

```
Float f1 = Float.valueOf("2.55");
Float f2 = Float.valueOf("4.5");

if(f1.compareTo(f2) == -1)
{
  System.out.println("2.55 < 4.5");
}
else
  System.out.println("No Tea");

//OUTPUT
2.55 < 4.5
```


C)

```
float t1 = 2.0f;
float t2 = 2.0f;
if(Float.compare(t1, t2) == 0)
{
  System.out.println("Float numbers are equal");
}

//OUTPUT
Float numbers are equal
```

D) All the above

Answer [=]

D

Explanation:
The methods Float.compare(f1,f2) and Float1.compareTo(Float2) works in the same way for Integer numbers also.

14) Choose the correct statement about Character wrapper class in Java?

A) Character class has static methods toUpperCase(char), toLowerCase(char)

B) Character class has static methods isDigit(char), isLetter(char)

C) Character class has static method valueOf(char) to convert char to Character.

D) All the above

Answer [=]

D

Explanation:
Example:


```
public class WrapperTest3
{
  public static void main(String[] args)
  {
    Character ct = Character.valueOf('a');
    char ch = ct.charValue();
    System.out.println(ch);
    System.out.println("Uppercase = " + Character.toUpperCase(ch));
    System.out.println("Is digit = " + Character.isDigit(ch));
    System.out.println("is Letter = " + Character.isLetter(ch));
  }
}

//OUTPUT
a
Uppercase = A
Is digit = false
is Letter = true
```

15) Choose the correct statement about Boolean wrapper class in Java?

A) Boolean class has static methods valueOf(boolean), valueOf("boolean") to convert boolean to Boolean object

B) Boolean class has static methods logicalAnd(boolean, boolean), logicalOr(boolean, boolean), logicalXor(boolean,boolean)

C) Boolean class has instance method equals() to check equality. compare() and compareTo() methods are also available

D) All the above

Answer [=]

D

Explanation:
Example:

```
public class WrapperTest4
{
  public static void main(String[] args)
  {
    Boolean b1 = Boolean.valueOf(false);
    Boolean b2 = Boolean.FALSE;
    System.out.println(b1 == b2);
    System.out.println(b1.equals(b2));
    System.out.println(Boolean.logicalAnd(true, false));
    System.out.println(Boolean.logicalOr(true, false));
    System.out.println(Boolean.logicalXor(true, false));
  }
}

//OUTPUT
true
true
false
true
true
```


16) Which exception is thrown for trying to create Wrapper class objects with wrong number type?

A) ArithmeticException

B) NumberFormatException

C) IllegarArgumentException

D) TypeNotPresentException

Answer [=]

B

Explanation:
Example with NumberFormatException:

```
public class WrapperTest5
{
  public static void main(String[] args)
  {
    Integer in = Integer.valueOf("0.0");
    System.out.println(in.intValue());
  }
}

OUTPUT:
java.lang.NumberFormatException: For input string: "0.0"
```

17) What is the output of the below code snipper with Short class object?

```
Short sh = Short.valueOf(10);
System.out.println(sh);
```

A) 10

B) 0

C) Compiler error

D) None

Answer [=]

C

Explanation:
The static method valueOf(short) accepts a short number but not integer. So, use casting to convert to short.

```
Error:
java.lang.Error: Unresolved compilation problem:
        The method valueOf(String) in the type Short is not applicable for the arguments (int)

//So, use casting.
Short sh = Short.valueOf((short)10);
```

18) Choose a correct statement about Type Casting using Wrapper class objects in Java?

A) You can not convert from Byte to Integer

B) You can not convert from Integer to Float

C) You can not convert from Integer to Long

D) All the above

Answer [=]

D

Explanation:
Yes. You can not convert one type of Wrapper object to another type Wrapper.

```
Error:

java.lang.Error: Unresolved compilation problem:
        Type mismatch: cannot convert from Byte to Integer
```

https://www.youtube.com/@codewitharrays

https://www.instagram.com/codewitharrays/

https://t.me/codewitharrays Group Link: https://t.me/ccee2025notes

+91 8007592194 +91 9284926333

codewitharrays@gmail.com

https://codewitharrays.in/project