**1. Which is true?**

A. "X extends Y" is correct if and only if X is a class and Y is an interface

B. "X extends Y" is correct if and only if X is an interface and Y is a class

C. "X extends Y" is correct if X and Y are either both classes or both interfaces

D. "X extends Y" is correct for all combinations of X and Y being classes and/or interfaces

**Answer: Option C**

**Explanation:** A is incorrect because classes implement interfaces, they don't extend them. B is incorrect because interfaces only "inherit from" other interfaces. D is incorrect based on the preceding rules.

# 2. Which of the following is true?

# 1. A class can extend more than one class. 2. A class can extend only one class but many interfaces. 3. An interface can extend many interfaces. 4. An interface can implement many interfaces. 5. A class can extend one class and implement many interfaces.

A. 1 and 2

B. 2 and 4

C. 3 and 5

D. 3 and 4

E. 2 and 5

**Answer: Option C**

# 3. What is the result of compiling and running the following code?

**class Base{**

**public Base(){**

**System.out.print("Base");**

**}**

**}**

**public class Derived extends Base{**

**public Derived(){**

**this("Examveda");**

**System.out.print("Derived");**

**}**

**public Derived(String s){**

**System.out.print(s);**

**}**

**public static void main(String[] args){**

**new Derived();**

**}**

**}**

A. ExamvedaDerived

B. ExamvedaBaseDerived

C. BaseExamvedaDerived

D. ExamvedaDerivedBase

E. Compilation Error

**Answer: Option C**

**Explanation:**

1. new Derived(); statement executes and invoke the non-parametrized constructor of derived class i.e.  
public Derived();  
2. As Derived class is a subclass of class Base so super(); executes and calls the super class constructor and prints **"Base"**.  
3. After that  
this("Examveda"); executes and call the parametrized constructor  
public Derived(String s); of Derived class as this always refer to the current object. So, it prints **"Examveda"**.  
4. Lastly the print statement executes and prints **"Derived"**  
Hence output is **BaseExamvedaDerived**.

# 4. What is the output of the following program code?

**abstract class C1{**

**public C1(){**

**System.out.print(1);**

**}**

**}**

**class C2 extends C1{**

**public C2(){**

**System.out.print(2);**

**}**

**}**

**class C3 extends C2{**

**public C3(){**

**System.out.println(3);**

**}**

**}**

**public class Test{**

**public static void main(String[] a){**

**new C3();**

**}**

**}**

A. 12

B. 23

C. 123

D. 321

**Answer: Option C**

# 5. The concept of multiple inheritance is implemented in Java by I.   Extending two or more classes. II.  Extending one class and implementing one or more interfaces. III. Implementing two or more interfaces.

A. Only (II)

B. (I) and (II)

C. (II) and (III)

D. Only (I)

E. Only (III)

**Answer: Option C**

# 6. What will be the output?

**interface A{**

**public void method1();**

**}**

**class One implements A{**

**public void method1(){**

**System.out.println("Class One method1");**

**}**

**}**

**class Two extends One{**

**public void method1(){**

**System.out.println("Class Two method1");**

**}**

**}**

**public class Test extends Two{**

**public static void main(String[] args){**

**A a = new Two();**

**a.method1();**

**}**

**}**

A. Compilation Error

B. Class One method1

C. Class Two method1

D. Throws a NoSuchMethodException at runtime.

E. None of these

**Answer: Option C**

# 7. What is the result of compiling and running this program?

**class Mammal{**

**void eat(Mammal m){**

**System.out.println("Mammal eats food");**

**}**

**}**

**class Cattle extends Mammal{**

**void eat(Cattle c){**

**System.out.println("Cattle eats hay");**

**}**

**}**

**class Horse extends Cattle{**

**void eat(Horse h){**

**System.out.println("Horse eats hay");**

**}**

**}**

**public class Test{**

**public static void main(String[] args){**

**Mammal h = new Horse();**

**Cattle c = new Horse();**

**c.eat(h);**

**}**

**}**

A. prints "Mammal eats food"

B. prints "Cattle eats hay"

C. prints "Horse eats hay"

D. Class cast Exception at runtime.

E. None of these

**Answer: Option A**

8.Determine output:

**class A{**

**public void method1(){**

**System.out.print("Class A method1");**

**}**

**}**

**class B extends A{**

**public void method2(){**

**System.out.print("Class B method2");**

**}**

**}**

**class C extends B{**

**public void method2(){**

**System.out.print("Class C method2");**

**}**

**public void method3(){**

**System.out.print("Class C method3");**

**}**

**}**

**public class Test{**

**public static void main(String args[]){**

**A a = new A();**

**C c = new C();**

**c.method2();**

**a = c;**

**a.method3(); // error is heare**

**}**

**}**

A. Class B method2 Class C method3

B. Class C method2 Class C method3

C. Compilation Error

D. Runtime exception

E. None of these

**Answer: Option C**

**Explanation:**

It is important to understand that it is the type of reference variable - not the type of the object that it refers to - that which determines what members can be accessed. That is, when a reference to a subclass object is assigned to a super class reference variable, we will have access only to those parts of the object defined by the superclass.  
In the above program method **method3()** is defined in the class **C** which is a subclass of **B** and so **A**. Even the reference variable a refers to c, a can't access **method3()** as this method is unknown to class **A**.

# 9. What will be printed after executing following program code?

**class Base{**

**int value = 0;**

**Base(){**

**addValue();**

**}**

**void addValue(){**

**value += 10;**

**}**

**int getValue(){**

**return value;**

**}**

**}**

**class Derived extends Base{**

**Derived(){**

**addValue();**

**}**

**void addValue(){**

**value += 20;**

**}**

**}**

**public class Test{**

**public static void main(String[] args){**

**Base b = new Derived();**

**System.out.println(b.getValue());**

**}**

**}**

A. 30

B. 10

C. 40

D. 20

E. None of these

**Explanation:**

When object of new **derived** is called, the flow goes to Derived() first, by default super(); is present in Derived() as the first statement, so the flow now goes to Base. Here value is initialised to 0 and then addValue() is called. The addValue has been overridden in Derived() hence The Base's addValue() will perform value+20(0+20) .After this control flows back to Derived()'s addValue() where again value+20 is done (20+20). Hence Answer is 40

# 10. What will be the output?

**class Parent{**

**public void method(){**

**System.out.println("Hi i am parent");**

**}**

**}**

**public class Child extends Parent{**

**protected void method(){**

**System.out.println("Hi i am Child");**

**}**

**public static void main(String args[]){**

**Child child = new Child();**

**child.method();**

**}**

**}**

A. Compiles successfully and print

B. Compiles successfully and print

C. Compile time error

D. Run Time error

E. None of This

**Answer: Option C**

**Explanation:**

We cannot reduce the visibility of the inherited method from super class. If the overridden or hidden method is public, then the overriding or hiding method must be public; otherwise, a compile-time error occurs. If the overridden or hidden method is protected, then the overriding or hiding method must be protected or public; otherwise, a compile-time error occurs. If the overridden or hidden method has default (package) access, then the overriding or hiding method must not be private; otherwise, a compile-time error occurs.

# 11. What will be the output?

**class One{**

**final int a = 15;**

**}**

**class Two extends One{**

**final int a = 20;**

**}**

**public class Test extends Two{**

**final int a = 30;**

**public static void main(String args[]){**

**Test t = new One();**

**System.out.print(t.a);**

**}**

**}**

A. 15

B. 20

C. 30

D. Compiler Error

E. None of these

**Answer: Option D**

**Explanation:**

We can't store super class object in subclass reference But we can store subclass object in super class reference.

# 12. What will be the output?

**class A{**

**int i = 10;**

**public void printValue(){**

**System.out.print("Value-A");**

**}**

**}**

**class B extends A{**

**int i = 12;**

**public void printValue(){**

**System.out.print("Value-B");**

**}**

**}**

**public class Test{**

**public static void main(String args[]){**

**A a = new B();**

**a.printValue();**

**System.out.print(a.i);**

**}**

**}**

A. Value-B 11

B. Value-B 10

C. Value-A 10

D. Value-A 11

E. None of these

**Answer: Option B**

**Explanation:**

If you create object of subclass with reference of super class like ( A a = new B();) then subclass method and super class variable will be executed.

# 13. What will be the result after compiling this code?

**class SuperClass{**

**public int doIt(String str, Integer... data)throws Exception{**

**String signature = "(String, Integer[])";**

**System.out.println(str + " " + signature);**

**return 1;**

**}**

**}**

**public class Test extends SuperClass{**

**public int doIt(String str, Integer... data){**

**String signature = "(String, Integer[])";**

**System.out.println("Overridden: " + str + " " +signature);**

**return 0;**

**}**

**public static void main(String... args){// throws Exception must be used**

**SuperClass sb = new Test();**

**sb.doIt("hello", 3);**

**}**

**}**

A. Overridden: hello (String, Integer[])

B. hello (String, Integer[])

C. Compilation fails

D. None of these

**Answer: Option C**

**Explanation:**

Exception must be caught or declared to be thrown.

14. **class A{**

**A(String s){}**

**A(){}**

**}**

**1. class B extends A{**

**2. B(){}**

**3. B(String s){**

**4. super(s);**

**5. }**

**6. void test(){**

**7. // insert code here**

**8. }**

**9. }**

# Which of the below code can be insert at line 7 to make clean compilation ?

A. A a = new B();

B. A a = new B(5);

C. A a = new A(String s);

D. All of the above

E. None of these

**Answer: Option A**

.

# 16. Determine output:

**class A{**

**public void printName(){**

**System.out.println("Name-A");**

**}**

**}**

**class B extends A{**

**public void printName(){**

**System.out.println("Name-B");**

**}**

**}**

**class C extends A{**

**public void printName(){**

**System.out.println("Name-C");**

**}**

**}**

**1. public class Test{**

**2. public static void main (String[] args){**

**3. B b = new B();**

**4. C c = new C();**

**5. b = c;**

**6. newPrint(b);**

**7. }**

**8. public static void newPrint(A a){**

**9. a.printName();**

**10. }**

**11. }**

A. Name B

B. Name C

C. Compilation fails due to an error on lines 5

D. Compilation fails due to an error on lines 9

E. None of these

**Answer: Option C**

**Explanation:**

# 17. What is the output for the below code ?

**class A{**

**private void printName(){**

**System.out.println("Value-A");**

**}**

**}**

**class B extends A{**

**public void printName(){**

**System.out.println("Name-B");**

**}**

**}**

**public class Test{**

**public static void main (String[] args){**

**B b = new B();**

**b.printName();**

**}**

**}**

A. Value-A

B. Name-B

C. Value-A Name-B

D. Compilation fails - private methods can't be override

E. None of these

**Answer: Option B**

**Explanation:**

You can not override private method , private method is not availabe in subclass . In this case printName() method a class A is not overriding by the printName() method of class B. printName() method of class B is a different method. So you can call printName() method of class B.

# 18. What will be the result of compiling and running the given code?

**class A{**

**int b=10;**

**private A(){**

**this.b=7;**

**}**

**int f(){**

**return b;**

**}**

**}**

**class B extends A{**

**int b;**

**}**

**public class Test{**

**public static void main(String[] args){**

**A a = new B();**

**System.out.println(a.f());**

**}**

**}**

A. Compilation Fails

B. Prints 0

C. Prints 10

D. Prints 7

E. None of these

**Answer: Option A**

## Solution(By Examveda Team)

Choice A is the correct answer.  
  
The code does not compile because the constructor of class A is declared as private. This creates a problem when the subclass constructor makes an implicit super() call to the parent class constructor at the time B is instantiated.  
  
Since the code does not compile, all the other choices are incorrect. If the constructor of A had not been private, the output would have been 7.

**Explanation:**

# 19. What will be the result of compiling and executing the following program code?

**class Vehicle{**

**public void printSound(){**

**System.out.print("vehicle");**

**}**

**}**

**class Car extends Vehicle{**

**public void printSound(){**

**System.out.print("car");**

**}**

**}**

**class Bike extends Vehicle{**

**public void printSound(){**

**System.out.print("bike");**

**}**

**}**

**public class Test{**

**public static void main(String[] args){**

**Vehicle v = new Car();**

**Bike b = (Bike) v;// run time error**

**v.printSound();**

**b.printSound();**

**}**

**}**

A. Compilation fails.

B. ClassCastException exception is thrown at runtime.

C. "vehiclecar" is printed.

D. "vehiclebike" is printed.

E. "carcar" is printed.

**Answer: Option B**

**Explanation:**