1. **Given the following piece of code:**

**public class School{**

**public abstract double numberOfStudent();**

**}**

**which of the following statements is true?**

A. The keywords public and abstract cannot be used together.

B. The method numberOfStudent() in class School must have a body.

C. You must add a return statement in method numberOfStudent().

D. Class School must be defined abstract.

**Answer: Option D**

# Which of the following class definitions defines a legal abstract class?

A. class A { abstract void unfinished() { } }

B. class A { abstract void unfinished(); }

C. abstract class A { abstract void unfinished(); }

D. public class abstract A { abstract void unfinished(); }

**Answer: Option C**

# Which of the following declares an abstract method in an abstract Java class?

A. public abstract method();

B. public abstract void method();

C. public void abstract Method();

D. public void method() {}

E. public abstract void method() {}

**Answer: Option B**

# Which of the following statements regarding abstract classes are true?

A. An abstract class can be extended.

B. A subclass of a non-abstract superclass can be abstract.

C. A subclass can override a concrete method in a superclass to declare it abstract.

D. An abstract class can be used as a data type.

E. All of the above

**Answer: Option E**

# Suppose A is an abstract class, B is a concrete subclass of A, and both A and B have a default constructor. Which of the following is correct? 1. A a = new A(); 2. A a = new B(); 3. B b = new A(); 4. B b = new B();

A. 1 and 2

B. 2 and 4

C. 3 and 4

D. 1 and 3

E. 2 and 3

**Answer: Option B**

# Which of the following is a correct interface?

A. interface A { void print() { } }

B. abstract interface A { print(); }

C. abstract interface A { abstract void print(); { }}

D. interface A { void print(); }

**Answer: Option D**

# Determine output of the following code.

**interface A { }**

**class C { }**

**class D extends C { }**

**class B extends D implements A { }**

**public class Test extends Thread{**

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

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

**if (b instanceof A)**

**System.out.println("b is an instance of A");**

**if (b instanceof C)**

**System.out.println("b is an instance of C");**

**}**

**}**

A. Nothing.

B. b is an instance of A.

C. b is an instance of C.

D. b is an instance of A followed by b is an instance of C.

**Answer: Option D**

# Given the following piece of code:

**public interface Guard{**

**void doYourJob();**

**}**

**abstract public class Dog implements Guard{ }**

# which of the following statements is correct?

A. This code will not compile, because method doYourJob() in interface Guard must be defined abstract.

B. This code will not compile, because class Dog must implement method doYourJob() from interface Guard.

C. This code will not compile, because in the declaration of class Dog we must use the keyword extends instead of implements.

D. This code will compile without any errors.

**Answer: Option D**

# In Java, declaring a class abstract is useful

A. To prevent developers from further extending the class.

B. When it doesn't make sense to have objects of that class.

C. When default implementations of some methods are not desirable.

D. To force developers to extend the class not to use its capabilities.

E. When it makes sense to have objects of that class.

**Answer: Option B**

# What will be the output?

**interface A{**

**public void method();**

**}**

**class One{**

**public void method(){**

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

**}**

**}**

**class Two extends One implements A{**

**public void method(){**

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

**}**

**}**

**public class Test extends Two{**

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

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

**a.method();**

**}**

**}**

A. will print Class One method

B. will print Class Two method

C. compiles fine but print nothing

D. Compilation Error

E. None of these

**Answer: Option B**

1. **interface Base{**

**boolean m1 ();**

**byte m2(short s);**

**}**

# which two code fragments will compile?

**1. interface Base2 implements Base {}**

**2. abstract class Class2 extends Base**

**{ public boolean m1(){ return true; }}**

**3. abstract class Class2 implements Base {}**

**4. abstract class Class2 implements Base**

**{ public boolean m1(){ return (7 > 4); }}**

**5. abstract class Class2 implements Base**

**{ protected boolean m1(){ return (5 > 7) }}**

A. 1 and 2

B. 2 and 3

C. 3 and 4

D. 1 and 3

E. 4 and 5

**Answer: Option C**

Explanation:

(3) is correct because an abstract class doesn't have to implement any or all of its interface's methods. (4) is correct because this method is correctly implemented ((7 > 4) is a boolean).  
  
(1) is incorrect because interfaces doesn't implement anything. (2) is incorrect because classes don't extend interfaces. (5) is incorrect because interface methods are implicitly public, so the methods being implemented must be public.

# Which two of the following are legal declarations for abstract classes and interfaces? 1. final abstract class Test {} 2. public static interface Test {} 3. final public class Test {} 4. protected abstract class Test {} 5. protected interface Test {} 6. abstract public class Test {}

A. 1 and 2

B. 2 and 4

C. 3 and 5

D. 5 and 6

E. 3 and 6

**Answer: Option E**

Explanation:

(3), (6). Both are legal class declarations.  
  
(1) is wrong because a class cannot be abstract and final "there would be no way to use such a class". (2) is wrong because interfaces and classes cannot be marked as static. (4) and (5) are wrong because classes and interfaces cannot be marked as protected.

1. **int p = 10; //line 1**

**public int q = 20; //line 2**

**public static int r = 30; //line 3**

**public static final int s = 40; //line 4**

**}**

# Which of the above line will give compilation error?

A. 1

B. 2

C. 3

D. 4

E. None of these

**Answer: Option E**

# What will happen after compiling this program code?

**abstract class MyClass{ //line 1**

**private int a, b;**

**public void call(int a, int b){**

**this.a = a;**

**this.b = b;**

**System.out.print(a+b);**

**}**

**}**

**public class Test{**

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

**MyClass m = new MyClass(); //line 2**

**m.call(12,25);**

**}**

**}**

A. Successful run and print 37

B. Compilation error due to line 1

C. Compilation error due to line 2

D. Runtime error

E. None of these

**Answer: Option C**

Explanation:

Abstract class is not concrete class which means object cannot be created for abstract class, its requires extending it and then create the object of extended class.

# Runnable is a \_\_\_\_\_ .

A. class

B. abstract class

C. interface

D. vaiable

E. method

**Answer: Option C**

# What is the output for the below code ?

**interface A{**

**public void printValue();**

**}**

**1. public class Test{**

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

**3. A a1 = new A(){**

**4. public void printValue(){**

**5. System.out.println("A");**

**6. }**

**7. };**

**8. a1.printValue();**

**9. }**

**10. }**

A. Compilation fails due to an error on line 3

B. A

C. Compilation fails due to an error on line 8

D. null

E. None of these

**Answer: Option B**

Explanation:

The A a1 reference variable refers not to an instance of interface A, but to an instance of an anonymous (unnamed) class. So there is no compilation error.

# What will be the output?

**1. public interface InfA{**

**2. protected String getName();**

**3. }**

**public class Test implements InfA{**

**public String getName(){**

**return "test-name";**

**}**

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

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

**System.out.println(t.getName());**

**}**

**}**

A. test-name

B. Compilation fails due to an error on lines 2

C. Compilation fails due to an error on lines 1

D. Compilation succeed but Runtime Exception

E. None of these

**Answer: Option B**

Explanation:

Illegal modifier for the interface method InfA.getName(); only public and abstracts are permitted.

# What will be the output for the below code ?

**public interface TestInf{**

**int i =10;**

**}**

**public class Test{**

**public static void main(String... args){**

**TestInf.i=12;**

**System.out.println(TestInf.i);**

**}**

**}**

A. Compile with error

B. 10

C. 12

D. Runtime Exception

E. None of these

**Answer: Option A**

Explanation:

All the variables declared in interface is implicitly static and final , therefore can't change the value.

# What will be the output when the following program is compiled and executed?

**abstract class TestAbstract{**

**String my\_name;**

**String myName(){**

**my\_name = "Examveda";**

**return my\_name;**

**}**

**abstract void display();**

**}**

**public class Test extends TestAbstract{**

**void display(){**

**String n = myName();**

**System.out.print("My name is "+ n);**

**}**

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

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

**t.display();**

**}**

**}**

A. Program will compile and execute successfully and prints

B. Compilation error as class can not be declared as abstract.

C. Program compiles but leads to runtime exception.

D. Compilation error occurs as the abstract class TestAbstract contains a non-abstract method.

E. None of these

**Answer: Option A**

Explanation:

The options B, C and D are incorrect options as in Java we can declare an abstract class comprising of abstract and non-abstract methods that will not lead to any compilation error. Therefore, option A is the correct answer implying that the 't' instance of Test class invokes the display method, which is implemented in the Test class. The display method invokes myName() method declared int the TestAbstract class and prints the name.

# What happens if the following program is compiled and executed?

**interface MyInterface{**

**void display();**

**}**

**interface MySubInterface extends MyInterface{**

**void display();**

**}**

**public class Test implements MySubInterface{**

**public void display(){**

**System.out.print("Welcome to Examveda.");**

**}**

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

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

**t.display();**

**}**

**}**

A. The code will lead to a compilation error as declaration of the display method has been provided in two interface.

B. The code will lead to a compilation error due to public modifier while declaring the display method.

C. The code will compile and execute successfully showing the output Welcome to Examveda.

D. The code will lead to a compilation error as the display method is not declared as abstract.

E. None of these

**Answer: Option C**

Explanation:

The program will compile and execute successfully as you can declare methods with same name in an interface and the method of either interface can be used, implying the option a is incorrect. The option B and D are incorrect as the methods of an interface are implicitly public and abstract.