Abstraction

1. Find Output

Which of the following is FALSE about abstract classes in Java
(A) If we derive an abstract class and do not implement all the abstract methods, then the derived class should also be marked as abstract using ‘abstract’ keyword
(B) Abstract classes can have constructors
(C) A class can be made abstract without any abstract method
(D) A class can inherit from multiple abstract classes.

Output

Answer: (D)

2. Find Output

Which of the following is true about interfaces in java.

1) An interface can contain following type of members. ....public, static, final fields (i.e., constants) ....default and static methods with bodies
2) An instance of interface can be created.
3) A class can implement multiple interfaces.
4) Many classes can implement the same interface.

Output

(A) 1, 3 and 4
(B) 1, 2 and 4
(C) 2, 3 and 4
(D) 1, 2, 3 and 4

Answer: (A)

3. Find Output

```
abstract class Ex
{
abstract void f1()
{
System.out.println("hello");
}
}
public class Test
{
public static void main(String[] args) {
        Ex e;
        e.f1();
    }
}
```

Output

a) Compilation error
b) Runtime error
c) Hello
d) No output

Ans: a

4. Find Output

```
abstract class Ex
{
abstract void f1()
{

};
}
public class Test
{
public static void main(String[] args) {
        Ex e;

    }
}
```

Output

a) Compilation error
b) Runtime error
c) f1()
d) No output

Ans: a

Explanation: abstract void f1() will raises compilation error

5. Find Output

```
abstract class Ex
{
abstract void f1();
}
public class Test
{
public static void main(String[] args) {
        Ex e = new Ex();
    }
}
```

Output

a) Compilation error
b) Runtime error
c) f1()
d) No output

Ans: a

6. Find Output

```
abstract class Ex
{
abstract void f1();
}
public class Test
{
public static void main(String[] args) {
        Ex e ;
    }
}
```

Output

a) Compilation error
b) Runtime error
c) f1()
d) No output

Ans: d

7. Find Output

```
abstract class Ex
{
abstract void f1();
}
class Ex1 extends Ex
{

}
public class Test
{
public static void main(String[] args) {
      Ex e ;
   }
}
```

Output

a) Compilation error
b) Runtime error
c) f1()
d) No output

Ans: a

8. Find Output

```
abstract class Ex
{



}
class Ex1 extends Ex
{

}
public class Test
{
public static void main(String[] args) {
        Ex1 e =new Ex1();
    }
}
```

Output

a) Compilation error
b) Runtime error
c) f1()
d) No output

Ans: d

9. Find Output

```
abstract class Ex
{
public static void main(String[] args) {
for(String s:args)
System.out.print(s);
}

}
class Ex1 extends Ex
{

}
public class Test
{
public static void main(String[] args) {
      Ex1 e =new Ex1();
      String s[] = {"raju","rani"};
      e.main(s);
   }
}
```

Output

a) rajurani
b) Compilation error
c) Runtime error
d) No output

Ans: a

10. Find Output

```
interface I1
{
void f1();
}
class Ex implements I1
{
void f1()
{
System.out.println("raju");
}
}
public class Test
{
public static void main(String[] args) {

   }
}
```

Output

a) raju
b) Compilation error
c) Runtime error
d) No output

Ans: b

11. Find Output

```
interface I1
{
void f1();
}
abstract class Ex implements I1
{

}
public class Test
{
public static void main(String[] args) {

   }
}
```

Output

a) f1()
b) Compilation error
c) Runtime error
d) No output

Ans: d

Find Output

Output