

JAVA MCQ'S IN INHERITANCE

1. Which of these keywords is used to refer to member of base class from a sub class?

- a) upper
- b) **super**
- c) this
- d) none of the mentioned

2. Which of these keyword used to inherit a class?

- a) super
- b) this
- c) extent
- d) **extends**

3. Which two classes use the Shape class correctly?

A. public class Circle implements Shape

```
{  
    private int radius;  
}
```

B. public abstract class Circle extends Shape

```
{  
    private int radius;  
}
```

C. public class Circle extends Shape

```
{  
    private int radius;  
    public void draw();  
}
```

D. public abstract class Circle implements Shape

```
{  
    private int radius;  
    public void draw();  
}
```

E. public class Circle extends Shape

```
{  
    private int radius;  
    public void draw()  
    {  
        /* code here */  
    }  
}
```

F. public abstract class Circle implements Shape

```
{  
    private int radius;  
    public void draw()  
    {  
        /* code here */  
    }  
}
```

- a) **B,E**
- b) A,C

JAVA MCQ'S IN INHERITANCE

- c) C,E
- d) T,H

4. What is the output of this program?

```
class A
{
    public int i;
    protected int j;
}
class B extends A
{
    int j;
    void display()
    {
        super.j=3;
        System.out.println(i + " " + j);
    }
}
class Output
{
    public static void main(String args[])
    {
        B obj = new B();
        obj.i=1;
        obj.j=2;
        obj.display();
    }
}
```

- a) 1 2
- b) 2 1
- c) 1 3
- d) 3 1

5. In order to restrict a variable of a class from inheriting to sub class, how variable should be declared?

- a) Protected
- b) **Private**
- c) Public
- d) Static

6. Static members are not inherited to sub class.

- a) True
- b) **False**

6. Which of the following is used for implementing inheritance through interface?

- a) inherited
- b) using

JAVA MCQ'S IN INHERITANCE

- c) extends
- d) **Implements**

7. Does Java support multiple level inheritances?

- a) **True**
- b) False

8. What is not type of inheritance?

- a) Single inheritance
- b) **Double inheritance**
- c) Hierarchical inheritance
- d) Multiple inheritance

9. **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)

10. 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();
    }
}
```

JAVA MCQ'S IN INHERITANCE

- A. Compilation Error
- B. Class One method1
- C. Class Two method1
- D. Throws a NoSuchElementException at runtime.
- E. None of these

11.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

12) Which is/are false statements

final class cannot be inherited

final method can be inherited

final method can be overridden

JAVA MCQ'S IN INHERITANCE

final variable of a class cannot be changed.

13) Java inheritance is used
for code re-usability
to achieve runtime polymorphism

Both of the above

None

14) Runtime polymorphism feature in java is

method overriding

method overloading

constructor overloading

operator overloading

15. Q) Runtime polymorphism feature in java is

method overriding

method overloading

constructor overloading

operator overloading