Java Abstraction Based Concepts Test:

1) **Below class ABC doesn't have even a single abstract method, but it has been declared as abstract. Is it correct?**

```
abstract class ABC
{
   void firstMethod()
   {
      System.out.println("First Method");
   }

   void secondMethod()
   {
      System.out.println("Second Method");
   }
}
```

2) **Why the below class is showing compilation error?**

```
abstract class AbstractClass
{
   abstract void abstractMethod()
   {
      System.out.println("First Method");
   }
}
```

3) **Which class is instantiable? Class A or Class B?**

```
abstract class A
{

}

class B extends A
{

}
```

4) **Below code snippet is showing compilation error? Can you suggest the corrections?**

```
abstract class A
{
   abstract int add(int a, int b);
}

class B extends A
{

}
```

5) **Is the following program written correctly? If yes, what value "result" variable will hold if you run the program?**

```
abstract class Calculate
{
   abstract int add(int a, int b);
}

public class MainClass
{
   public static void main(String[] args)
   {
      int result = new Calculate()
      {
         @Override
         int add(int a, int b)
         {
            return a+b;
         }
      }.add(11010, 022011);
   }
}
```

6) **Can we write explicit constructors in an abstract class?**
7) **Can you identify the error in the below code?**

```
abstract class AbstractClass
{
   private abstract int abstractMethod();
```

```
}
```

8) **Can we declare protected methods in an interface?**

9) **What will be the output of the following program?**

```
abstract class A
{
   abstract void firstMethod();

   void secondMethod()
   {
      System.out.println("SECOND");

      firstMethod();
   }
}

abstract class B extends A
{
   @Override
   void firstMethod()
   {
      System.out.println("FIRST");

      thirdMethod();
   }

   abstract void thirdMethod();
}

class C extends B
{
   @Override
   void thirdMethod()
   {
      System.out.println("THIRD");
   }
}

public class MainClass
{
   public static void main(String[] args)
   {
      C c = new C();

      c.firstMethod();
```

```
        c.secondMethod();

        c.thirdMethod();
    }
}
```

10) What will be the output of the below program?

```
abstract class X
{
    public X()
    {
        System.out.println("ONE");
    }

    abstract void abstractMethod();
}

class Y extends X
{
    public Y()
    {
        System.out.println("TWO");
    }

    @Override
    void abstractMethod()
    {
        System.out.println("THREE");
    }
}

public class MainClass
{
    public static void main(String[] args)
    {
        X x = new Y();

        x.abstractMethod();
    }
}
```

11) Can we declare abstract methods as static?

12) Is the below program written correctly? If yes, what will be the output?

```
abstract class A
{
   {
      System.out.println("AAA");
   }
}

abstract class B extends A
{
   {
      System.out.println("BBB");
   }
}

class C extends B
{
   {
      System.out.println("CCC");
   }
}

public class MainClass
{
   public static void main(String[] args)
   {
      C c = new C();
   }
}
```

13) What will be the output of the following program?

```
abstract class A
{
   abstract int firstMethod(int i);

   abstract int secondMethod(int i);

   int thirdMethod(int i)
   {
      return secondMethod(++i);
   }
}
```

```
abstract class B extends A
{
   @Override
   int secondMethod(int i)
   {
      return firstMethod(++i);
   }
}

class C extends B
{
   @Override
   int firstMethod(int i)
   {
      return ++i;
   }
}

public class MainClass
{
   public static void main(String[] args)
   {
      C c = new C();

      System.out.println(c.thirdMethod(121121));
   }
}
```

14) Can we keep static initialization blocks inside an abstract class?

15) Is the below program written correctly? If yes, what will be the output?

```
abstract class XYZ
{
   {
      System.out.println(1);
   }

   public XYZ()
   {
      System.out.println(2);

      abstractMethod();
   }
```

```
    abstract void abstractMethod();
}

class PQR extends XYZ
{
    {
        System.out.println(3);
    }

    public PQR()
    {
        System.out.println(4);
    }

    @Override
    void abstractMethod()
    {
        System.out.println(5);
    }
}

public class MainClass
{
    public static void main(String[] args)
    {
        PQR pqr = new PQR();
    }
}
```

16) Can you identify the error in the below code?

```
class X
{
    public X()
    {
        System.out.println("Constructor One");
    }

    abstract X(int i)
    {
        System.out.println("Constructor Two");
    }
}
```

17) Abstract methods can be declared as final. True or False?

18) Is the below code written correctly?

```
class X
{
   abstract class Y
   {
      class Z
      {

      }
   }
}
```

19) What will be the output of the following program?

```
class ClassOne
{
   int methodOne(int i, int j)
   {
      return i++ + ++j - ++i - j++;
   }
}

abstract class ClassTwo extends ClassOne
{
   abstract int methodOne(int i, int j, int k);

   @Override
   int methodOne(int i, int j)
   {
      return methodOne(i, j, i+j);
   }
}

class ClassThree extends ClassTwo
{
   @Override
   int methodOne(int i, int j, int k)
   {
      return --i - j-- + ++k - i++ + ++j - k--;
   }
}

public class MainClass
```

```
{
   public static void main(String[] args)
   {
      ClassOne one = new ClassOne();

      ClassThree three = new ClassThree();

      System.out.println(three.methodOne(one.methodOne(10101, 20202),
one.methodOne(20202, 10101)));
   }
}
```

20) Is the below code written correctly?

```
class A
{
   void methodOfA()
   {
      abstract class B
      {

      }
   }
}
```