1. What is Abstraction in Java?

Abstraction is a process of hiding the implementation details and showing only functionality to the user.

```
package coding;
abstract class Sunstar {
        int age = 21;
        float salary = 222.2F;
   abstract public void printInfo();
   public void salary() {
       System.out.println("Salary:"+salary);

   }
}

class Employee extends Sunstar {
  public void printInfo() {
       String name = "avinash";

       System.out.println("Name:"+name);
       System.out.println("Age:"+age);

   }

}
public class Abstract1 {
   public static void main(String[] args) {
       Employee a = new Employee();
            a.printInfo();
            a.salary();
       }

}
```

OUTPUT:
Name: ILAKKIYA
Age:21
Salary:222.2

2. How to achieve or implement Abstraction in Java?

Using abstract classes(0% to 100%):

Abstract classes achieve partial abstraction as concrete methods can also be defined in them.

Program:

```
package coding;

abstract class A{
	int a =10;
	int b=5;
	abstract public void add();
	public void sub() {
		int c=a-b;
		System.out.println("Sub:"+c);


	}
}
class B extends A{
	public void add() {
		int c = a+b;
		System.out.println("Add:"+c);

	}
}
public class Abstraction {

	public static void main(String[] args) {
		B b = new B();
		b.add();
		b.sub();

	}
```

```
}
```

OUTPUT:
10
5

Using interfaces(100%):
Interfaces achieve complete abstraction as only abstract methods can be defined in them.
Program:

```
package coding;

abstract class A{
	int a =10;
	int b=9;
	abstract public void add();
	public void sub() {
		int c=a-b;
		System.out.println("Sub:"+c);


	}
}
class B extends A{
	public void add() {
		int c = a+b;
		System.out.println("Add:"+c);

	}
}
public class Abstraction {

	public static void main(String[] args) {
		B b = new B();
		b.add();
		b.sub();
```

```
        }

}
```

OUTPUT:
10
1

3. What is Abstract class in Java? How to define it?

A class which is declared with the abstract keyword is known as an abstract class in Java. It can have abstract and non-abstract methods (method with the body).

```
package demo2;
abstract class B {
   abstract public void fun();
}


class D extends B {
  public void fun()
   {
      System.out.println("Derived class");
   }
}

public class Abstract {
 public static void main(String[] args) {
                B b = new D();
      b.fun();

       }
```

```
}
```

OUTPUT:
Derived

4 .What is the difference between abstract class and concrete class?

Abstract class can have both an abstract as well as concrete methods. A concrete class can only have concrete methods. Even a single abstract method makes the class abstract.

Abstract class can not be instantiated using new keyword. Concrete class can be instantiated using new keyword.

Abstract class may or may not have abstract methods. Concrete clas can not have an abstract method.

Abstract class can not be declared as a final class. Concrete class can be declared final.

ABSTRACT CLASS:

```
package coding;

abstract class A{
	int id=01;
	abstract public void add();
	public void details() {
		String name = "Ilakkiya";
		System.out.println(name);



	}
}
class B extends A{
	public void detail() {
```

```
            System.out.println(id);}
}
public class Abstraction {

        public static void main(String[] args) {
                B b = new B();
                b.detail();
                b.details();

        }

}
```

CONCRETE CLASS:

```
package coding;

class Add{
        int id =1;
    String name="ilakkiya";

    public void details(){
        System.out.println(id);
        System.out.println(name);
         }
}


public class Concrete {

        public static void main(String[] args) {
                Add a = new Add();
                a.details();

        }
```

```
}
```

5. What is Abstract in Java?

An abstract is a non-access modifier in java applicable for classes, and methods but not variables.
Abstract class is a restricted class that cannot be used to create objects (to access it, it must be inherited from another class).
Abstract method can only be used in an abstract class, and it does not have a body.

```
package coding;

abstract class A{
	int id=01;
	abstract public void add();
	public void details() {
		String name = "Ilakkiya";
		System.out.println(name);



	}
}
class B extends A{
	public void detail() {
		System.out.println(id);
		}
}

public class Abstraction {

	public static void main(String[] args) {
		B b = new B();
		b.detail();
		b.details();
```

```
        }

}
```

OUTPUT:
1
Ilakkiya

6. Can abstract modifier applicable for variables?

Abstract Access Modifier is a modifier applicable only for classes and methods but not for variables.

```
abstract class A{
   abstract int a=10;//not declare

   abstract public void a() ;
}
```

7. What is Abstract method in Java?

Abstract class contains abstract method and non abstract method.
A method declared using the **abstract** keyword within an abstract class and does not have a definition (implementation) is called an abstract method.

```
package coding;

abstract class Company{
   String name="Ilakkiya";
   abstract public void names() ;//abstract method

}
class Details extends Company{
        public void names() {
                System.out.println(name);
```

```
        }
}
```

```
public class Variable {

        public static void main(String[] args) {
        Details d = new Details();
        d.names();

        }

}
```

8. Can an abstract method be declared as static?

An abstract method cannot be static.

Program:

```
package coding;

abstract class Company{
    String name="Ilakkiya";
   static abstract public void names() ;//can't declare

}
```

9. Can an abstract method be declared with private modifier?

you cannot use it from the same class, you need to override it from subclass and use. *Therefore, the abstract method cannot be private.*

Program:

```
package coding;

abstract class Company{
    String name="Ilakkiya";
```

```
  private abstract public void names() ;

}
```

10.What is Concrete method in Java?

A concrete method means, the method has complete definition but it can be overridden in the inherited class. If we make this method "final" then it can not be overriden. Declaring a method or class "final" means its implementation is complete.

PROGRAM:

```
package coding;

abstract class Company{
	String name="Ilakkiya";
	public void names() {
		System.out.println(name);
	}
}

class Details extends Company{

}


public class Variable {

	public static void main(String[] args) {
	Details d = new Details();
	d.names();

	}

}
```

11. When to use Abstract class in Java?

If we are using the inheritance concept since it provides a common base class implementation to derived classes. You want to define common methods and fields that will be shared among multiple related classes. You want to enforce that certain methods must be implemented by any concrete (sub)class that extends the abstract class. You want to provide a level of abstraction and hide the underlying implementation details from the user of the class.

```
package coding;

abstract class Company{
	int id =1;
	String name="Ilakkiya";
	String dob="17-01-2001";
	abstract public void names();
	abstract public void detail();
	public void details() {
		System.out.println(id);
	}

}

class Details extends Company{
	public void names() {
	System.out.println(name);
}
	public void detail() {
		System.out.println(dob);
	}

}


public class Variable {

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

```
        Details d = new Details();
    d.details();
        d.names();
        d.detail();

        }

}
```

12.When to use Abstract method in Java?

The abstract Method is used for creating blueprints for classes or interfaces. Here methods are defined but these methods don’t provide the implementation. Abstract Methods can only be implemented using subclasses or classes that implement the interfaces.

```
package coding;
abstract class Company{
        int id =1;
        String name="Ilakkiya";
        String dob="17-01-2001";
        abstract public void names();
        abstract public void detail();
        public void details() {
                System.out.println(id);
        }

}

class Details extends Company{
        public void names() {
        System.out.println(name);
}
        public void detail() {
```

```
            System.out.println(dob);
        }

}


public class Variable {

        public static void main(String[] args) {
        Details d = new Details();
    d.details();
        d.names();
        d.detail();

        }

}
```

13. Is abstract class a pure abstraction in Java?

Abstract class may or may not have abstract methods, so it is not a pure abstraction.

14. Is it possible to create an object of abstract class in Java?

We cannot create objects of an abstract class. To implement features of an abstract class, we inherit subclasses from it and create objects of the subclass. A subclass must override all abstract methods of an abstract class. However, if the subclass is declared abstract, it's not mandatory to override abstract methods.

Program:

```
abstract class A{
abstract public void m1();
}
Public class B{
Public static void main(String [] args){
A a = new A ();\\ abstract class cannot create object directly.
```

```
}
}
```

15. Is it possible that an abstract class can have without any abstract method?

yes, you can declare abstract class without defining an abstract method in it.

```
package coding;

abstract class A{
	//not abstract method

	public void details() {
		String name = "Ilakkiya";
		System.out.println(name);



	}
}
class B extends A{

}

public class Abstraction {

	public static void main(String[] args) {
		B b = new B();
	   b.details();

	}

}
```

PROGRAM EXPLANATION WITH OUTPUT

a) abstract class A
{
void m1();
}

Ans: An Abstract method must be declare within the abstract keyword and contains only declaration (not body). So this method not declare abstract keyword .so It comes error.

Program:

```
package abst;
abstract class Animals{
	abstract public void dog();
}
class Animals1 extends Animals{
	public void dog() {
		System.out.println("Dog is a pet animal.");
	}
}
public class Method {

	public static void main(String[] args) {
		Animals1 a = new Animals1();
		a.dog();


	}

}
```

Output:
Dog is a pet animal.

Correction:

Abstract method must be declared with abstract keyword and not body.so declare abstract keyword to method

b) public class A {
abstract void m1();
}

Ans: Abstract class can declare abstract method and non abstract method. But Non abstract class can't decalare abstract method. So it is error.

Program:

```
package abst;
abstract class Animals{
	abstract public void cat();
}
class Animals1 extends Animals{
	public void cat() {
		System.out.println("Cat is a pet animal.");
	}
}
public class Method {

	public static void main(String[] args) {
		Animals1 a = new Animals1();
		a.cat();


	}

}
```

Output:

Cat is a pet animal.

Correction:

Abstract method only declare in abstract class.so create abstract class .

c) abstract public class A {
abstract void m1();
}
Ans: This syntax is wrong. Because abstract class cann't contains access modifiers.
Program:

```
package abst;
abstract class Add{
	int a =10;
	int b=5;
	abstract public  void add();
}
class Add1 extends Add{
	public void add() {
		int c =a+b;
		System.out.println("Addition:"+c);
	}
}
public class Addition {

	public static void main(String[] args) {
		Add1 s = new Add1();
		s.add();

	}

}
```

Output:
Addition: 15
Correction:
Abstract class with access modifiers is illegal modifier.so delete public access modifier .

d) abstract public class A {
void m1() { }
}
Ans: Abstract class does not contains access modifiers and non access modifers. So it is error.
Another way: remove abstract keyword in abstract class then it is automatic change to concrete class.
Program:

```
package abst;
 class Add{
   int a=10;
   int b=10;
         public void add() {
                        int c =a+b;
                        System.out.println("Addition:"+c);
                }
        }


public class Addition {

        public static void main(String[] args) {
        Add a = new Add();
        a.add();


        }

}
```

Output:
Addition:20

Correction:
Remove access modifier and abstract keyword.

e) public abstract class A {
abstract void m1();
A(){ }
void m2() { }
}

Ans: Abstract class cannot contain access modifiers and non access modifiers.
Program:

```
package abst;
 abstract class Add{
   abstract public void  walk();
   Add(){
       System.out.println("He is jumping.");
   }
        public void run() {
                System.out.println("she is running.");

               }
       }
 class Add1 extends Add{
        public void walk() {
                System.out.println("He is walking.");
         }
 }


public class Addition {

       public static void main(String[] args) {
       Add1 a = new Add1();
       a.walk();
       a.run();
```

```
        }

}
```

Output:

He is jumping.
He is walking.
she is running.

Correction:

Remove access modifier in abstract class.

f) public abstract class A {
abstract int x = 100;
abstract void m1();
abstract void m2();
}

Ans: An abstract class can't declare access modifier. Then variable can't declare abstract keyword.

Program:

```
package abst;
 abstract class Add{
    int a =10;
    abstract public void  walk();
    abstract public void run();
        }
 class Add1 extends Add{
        public void walk() {
                System.out.println("First abstract method .");
System.out.println(a);

        }
        public void run() {
                System.out.println("Second abstract method.");
        }
```

```
 }


public class Addition {

        public static void main(String[] args) {
        Add1 a = new Add1();
        a.walk();
        a.run();


        }

}
```

Output:
First abstract method.
10
Second abstract method.

Correction:
Remove abstract keyword in abstract variable.

g) public abstract class A {
abstract void m1();
}
public class Test {
public static void main(String[] args) {
A a = new A();
}
}
Ans:Abstract class can't contains access modifiers. Then abstract class must be inherit subclass and create object in this subclass.
An Abstract class can't create object.

Program:

```
package abst;

abstract class A{
	abstract void m1();

}
	class B extends A{
		void m1() {
			System.out.println("It is subclass.");
		}
	}

public class Constructor {

	public static void main(String[] args) {
		  B obj = new B();
		obj.m1();
	}

	}
```

Output:
It is subclass.

Correction:
Create sub class and create object of subclass.

h) public abstract class A {
abstract void m1();
A(){ }
static void m2() {System.out.println("Hello Java!"); }
}
public class B extends A {
void m1(){
A.m2();

```
}
}
```

Ans: Remove access modifier . First execute constructor part then m2 execute. So Hello java is printed.

Program:

```
package abst;

abstract class A1{
        abstract public void dog();
    A1() {
                System.out.println("It is a cat.");
        }
    static void m2() {
         System.out.println("Hello java!");
    }
}
class B1 extends A1{
        public void dog() {
                System.out.println("It is a dog.");

        }
}

public class Variable {

        public static void main(String[] args) {
                A1 b = new B1();
                b.dog();
                A1.m2();


        }
}
```

Output:

It is a cat.

It is a dog.
Hello java!

Correction:

Remove access modifier in abstract class.

i) public abstract class A {
abstract void m1();
private A(){ }
}
public class B extends A { }

Ans: Private constructors in abstract classes are used for control: Private constructors in abstract classes are typically used for controlling the instantiation of the abstract class itself. By making the constructor private, you ensure that no instances of the abstract class can be created directly

Program:

```
public abstract class MyAbstractClass {
    private MyAbstractClass() {
        // Private constructor to prevent direct instantiation
    }

    public abstract void someMethod();

    public static MyAbstractClass createInstance() {
        return new MyConcreteSubclass();
    }
}

public class MyConcreteSubclass extends MyAbstractClass {
    @Override
```

```
    public void someMethod() {
        System.out.println("Concrete subclass method.");
    }
}

public class Main {
    public static void main(String[] args) {
        MyAbstractClass instance = MyAbstractClass.createInstance();
        instance.someMethod();
    }
}
```