**Q. Can we overload the main method?**

Yes

**Q. A Java Constructor returns a value but, what?**

No. Java constructor cannot return a value. If required, just create a method which calls the required constructor and returns the required value.

**Q. Can we create a program without main method?**

Yes You **can** compile and execute **without main method** By using static block

**Q. What are the six ways to use this keyword?**

* this can be used to get the current object.
* this can be used to invoke current object's method.
* this() can be used to invoke current class constructor.
* this can be passed as a parameter to a method call.
* this can be passed as a parameter to a constructor.
* this can be used to return the current object from the method

**Q. Why is multiple inheritance not supported in Java?**

**Java supports multiple inheritance** through interfaces only. A class can implement any number of interfaces but can extend only one class. **Multiple inheritances** is **not supported** because it leads to deadly diamond problem.

The problem with multiple inheritance is that two classes may define different ways of doing the same thing, and the subclass can't choose which one to pick.

**Q. Why use aggregation?**

When an object A contains a reference to another object B or we can say Object A has a HAS-A relationship with Object B, then it is termed as Aggregation.

Aggregation helps in reusing the code. Object B can have utility methods and which can be utilized by multiple objects. Whichever class has object B then it can utilize its methods.

**Q. Can we override the static method?**

No, you can not **override static method** in Java

**Q. What is threat in java?**

**Thread** is a light weight process which helps in running the tasks in parallel. The **threads** works independently and provides the maximum utilization of the CPU

**Q. What is the covariant return type?**

**Covariant return type** refers to **return type** of an overriding method. It allows to narrow down **return type** of an overridden method without any need to cast the **type** or check the **return type**. **Covariant return type** works only for non-primitive **return types**.

**Q. What are the three usages of Java super keyword?**

* **super** variable refers immediate parent class instance.
* **super** variable can invoke immediate parent class method.
* **super**() acts as immediate parent class constructor

**Q. Why use instance initializer block?**

**Instance Initializer block** is **used** to initialize the **instance** data member. It run each time when object of the class is created.

**Q. Can we initialize blank final variable?**

Yes! You **can initialize** a **blank final variable** in constructor or instance **initialization** block.

**Q. What is the usage of a blank final variable?**

A final variable that is not initialized at the time of declaration is known as blank final variable.

**Q. What is a marker or tagged interface?**

A **marker interface** is an **interface** that has no methods or constants inside it. It provides run-time type information about objects, so the compiler and JVM have additional information about the object.

**Q. What is runtime polymorphism or dynamic method dispatch?**

**Dynamic method dispatch** is the mechanism by which a call to an overridden method is resolved at **run time**, rather than compile time.

**Q. What is the difference between static and dynamic binding?**

S.B -> Binding happens during compile time. Ex: method overloading

D.B -> Binding happens during run time. Ex: method overriding

**Q. How downcasting is possible in Java?**

When Subclass type refers to the object of Parent class, it is known as **downcasting**.

**Q. What is the purpose of a private constructor?**

Making something **private** doesn't mean nobody can access it. It just means that nobody outside the class can access it. So **private constructor** is useful too.

**Q. What is object cloning?**

**Object cloning** refers to creation of exact copy of an **object**

**Q. Advantage of OOPs**

* Improved software-development productivity
* Improved software maintainability
* Faster development
* Lower cost of development
* High quality software

**Q. Which oop concept is used as reuse mechanism?**

Inheritance

**Q. Which keyword can be used for overloading?**

Operator keyword is used for overloading

**Q. Can static method use non static members?**

No/ False

**Q. How many instances can be created for an abstract class?**

Zero