JAVA overriting vs overloading

In Java, method overloading occurs when two or more methods in the same class have the same method name but a different number or type of parameters. Method overriding occurs when a subclass provides a specific implementation of a method that is already provided by its superclass.

JAVA static field vs non static field

In Java, a static field is a field that belongs to the class rather than an instance of the class. This means that there is only one copy of the static field shared among all instances of the class. A non-static field, also known as an instance field, is a field that belongs to a specific instance of the class. This means that each instance of the class has its own copy of the non-static field.
Static fields are typically used to store information that is shared among all instances of a class, such as a counter for the number of instances created. Non-static fields are used to store information that is specific to a single instance of a class, such as the name of a person.

JAVA what is abstract class

An abstract class in Java is a class that cannot be instantiated, but can be subclassed. It is used as a base class for other classes that implement the functionality declared by the abstract class. An abstract class can have abstract and non-abstract methods (abstract methods do not have a body).
You can define an abstract class by using the "abstract" keyword before the class keyword.

JAVA interface vs class

In Java, an interface is a collection of abstract methods, while a class is a blueprint for creating objects.
An interface defines a contract for a group of related classes, specifying a set of methods that these classes must implement. Interfaces cannot be instantiated, they can only be implemented by classes. A class can implement multiple interfaces, but can only inherit from one superclass.
A class, on the other hand, can have both concrete (implemented) and abstract (not implemented) methods, fields, constructors and can also have static and non-static methods, fields. A class can also be extended and instantiated.
An interface provides a way to achieve multiple inheritance and abstraction, while a class is used for implementation and instantiation.

refference type vs primitive type java

In Java, there are two types of data types: primitive and reference.
Primitive data types are the basic data types that are built into the Java language. They include:

- boolean
- byte
- char
- short
- int
- long
- float
- double
- These data types hold the actual value, and are not objects. They are stored on the stack and are faster to access than reference types.

Reference data types, on the other hand, are objects that are stored on the heap. They include:

- Arrays
- Classes
- Interfaces
- Enumerations
- Strings

Reference types are created by the programmer and are stored on the heap, and they hold a reference to the memory address where the object is stored. These types are slower to access than primitive types because they require additional memory accesses to retrieve the actual value.
You can tell the difference between a primitive and reference type by the way it is declared and used. Primitive types are typically declared using keywords like int, float, etc., and are used to store simple values. Reference types are typically declared using a class or interface name, and are used to store references to objects.

static vs non static methods

In Java, a static method is a method that can be called directly on a class, without the need to create an instance of the class. A non-static method, also known as an instance method, can only be called on an instance of the class.
Static methods are typically used to perform operations that do not depend on the state of a specific object. For example, a utility class might have only static methods that perform mathematical operations or string manipulation.
Non-static methods, on the other hand, are used to perform operations that depend on the state of a specific object. For example, a class that represents a bank account might have non-static methods for depositing and withdrawing money.

checked vs unchecked exceptions

In Java, checked exceptions are exceptions that are checked by the compiler at compile-time, while unchecked exceptions are not checked by the compiler.
Checked exceptions are those that are derived from the Exception class, except for the RuntimeException and its subclasses. Examples of checked exceptions include IOException, SQLException, and ClassNotFoundException. Because these exceptions are checked by the compiler, any method that might throw a checked exception must either catch it or declare it in its throws clause. This means that the developer must anticipate and handle these exceptions in the code.
Unchecked exceptions are those that are derived from the RuntimeException class and its subclasses. Examples of unchecked exceptions include NullPointerException, IllegalArgumentException, and ArithmeticException. Because these exceptions are not checked by the compiler, a method is not required to catch or declare them. The developer is not required to anticipate and handle these exceptions in the code. However, it is good practice to do so, as they indicate a bug in the code.
In general, checked exceptions are used to indicate a situation that can be handled, but the handling must be done by the developer, while unchecked exceptions are used to indicate a situation that is caused by a bug in the code and should be fixed by the developer.

interfaces vs abstract classes

In Java, an interface is a collection of abstract methods, while an abstract class is a class that cannot be instantiated, but can be subclassed.
An interface defines a contract for a group of related classes, specifying a set of methods that these classes must implement. Interfaces cannot have any implementation and cannot have any state, they can only have abstract methods, constants and default methods. A class can implement multiple interfaces, but can only inherit from one superclass.
An abstract class, on the other hand, can have both abstract and non-abstract methods, fields, constructors and can also have static and non-static methods, fields. An abstract class can also be extended, but cannot be instantiated.