1-Why does defining a custom constructor suppress the default constructor in C#?

- **Default constructor (parameterless)** is auto-generated only if you don't define any constructor.
- Once you define a custom constructor, the compiler **stops generating** the default one.
- Reason: to avoid ambiguity, enforce your construction rules, and give you full control over object creation.
- If you still want a parameterless constructor, you must **explicitly write it**.

2-How does method overloading improve code readability and reusability?

Improves Reusability:

- The same method can work with different inputs without duplicating code.
- Easy to extend by adding new overloads in the future.

3-What is the purpose of constructor chaining in inheritance?

Purpose of Constructor Chaining in Inheritance – Summary

- Ensure the **base class is initialized first** before the derived class.
- Promotes **code reusability** by avoiding duplicate initialization code.
- Maintains **consistency** so all child objects start with a properly set base.

- Provides **extensibility**, since changes in the base constructor automatically apply to derived classes.

4-How does new differ from override in method overriding?

Summary: new vs override in C#

- **override** → Replaces a virtual/abstract method in the base class, supports **polymorphism** (child method runs even when accessed via base reference).
- **new** → Hides a method in the base class, does **not** support polymorphism (base method runs when accessed via base reference).

5-Why is ToString() often overridden in custom classes?

- By default, ToString() (inherited from System.Object) only returns the class name.
- We override it to provide a **meaningful, human-readable description** of the object.
- Benefits:
    1. Easier debugging and logging.
    2. Clearer representation of the object's state.
    3. More user-friendly output when printing objects.

In short: **Overriding ToString() makes objects more descriptive and useful instead of just showing the class name.**