Part01

1. **Why does defining a custom constructor suppress the default constructor in C#?**
    - Because once you define any constructor explicitly, the compiler does not generate the default parameterless constructor automatically.

---

2. **How does method overloading improve code readability and reusability?**
    - It allows using the same method name for different parameter lists, making the code easier to read and reuse without remembering multiple method names.

---

3. **What is the purpose of constructor chaining in inheritance?**
    - To ensure that the base class constructor is called to initialize base class members before executing the derived class constructor.

---

4. **How does new differ from override in method overriding?**
    - new hides the base class method without affecting polymorphism, while override replaces the base method's implementation and supports polymorphic behavior.

---

5. **Why is ToString() often overridden in custom classes?**
    - To provide a meaningful string representation of the object instead of the default class name output.

---

6. **What is the difference between a virtual method and an abstract method in C#?**
    - A **virtual method** has a default implementation but can be overridden.
    - An **abstract method** has no implementation and must be overridden in derived classes.

Part02

✅ **Difference between class and struct in C#:**

1. **Type**:
    - **Class** → Reference type (stored on the heap).
    - **Struct** → Value type (stored on the stack).
2. **Inheritance**:
    - **Class** → Supports inheritance.
    - **Struct** → Does **not** support inheritance (but can implement interfaces).
3. **Default Constructor**:
    - **Class** → Can define a parameterless constructor.
    - **Struct** → Cannot define a custom parameterless constructor (compiler provides one automatically).
4. **Nullability**:
    - **Class** → Can be null.
    - **Struct** → Cannot be null (unless defined as Nullable<T> or T?).
5. **Performance**:
    - **Class** → Slower access (heap allocation, garbage collection).
    - **Struct** → Faster for small, lightweight objects because stored on stack.

✅ **Relations between classes (other than inheritance):**

1. **Association**:
    - A general relationship where one class uses another.
    - Example: Teacher uses Student.
2. **Aggregation** (Has-a):
    - A "whole–part" relationship, but the part can exist independently.
    - Example: Department has Teachers, but teachers can exist without the department.
3. **Composition** (Strong Has-a):
    - Stronger form of aggregation where the part cannot exist without the whole.
    - Example: Car has an Engine. If the car is destroyed, the engine is too.
4. **Dependency** (Uses-a):
    - A temporary relationship where one class depends on another to perform a function.
    - Example: Report depends on Printer to print.

📌 So:

- **Inheritance** → "is-a" relationship.
- **Aggregation/Composition** → "has-a" relationship.
- **Association** → "works-with" relationship.
- **Dependency** → "uses" relationship.

Part03 Bonus

what is static and dynamic binding?

Static Binding (Early Binding)

- Resolved **at compile-time**.
- Used in **method overloading / non-virtual methods**.
- **Faster**.

Dynamic Binding (Late Binding)

- Resolved **at runtime**.
- Used in **method overriding (virtual/override)**.
- **More flexible** but slower.