The core creational design patterns are Factory Method, Abstract Factory, Builder, Prototype, and Singleton, which provide mechanisms for creating objects, enhancing flexibility and code reuse.

Here's a breakdown of each:

* **Factory Method:**

Defines an interface for creating an object, but lets subclasses decide which class to instantiate. This pattern centralizes object creation, allowing subclasses to change the type of objects created.

* **Abstract Factory:**

Provides an interface for creating families of related or dependent objects without specifying their concrete classes. This pattern is useful when you need to create multiple types of objects that are related, and you want to ensure that the correct types are created together.

* **Builder:**

Separates the construction of a complex object from its representation, allowing the same construction process to create different representations. This pattern is useful when you need to create complex objects with many optional parts, and you want to avoid having to pass a lot of parameters to the constructor.

* **Prototype:**

Specifies the kind of objects to create using a prototypical instance, and creates new objects by cloning this prototype. This pattern is useful when you need to create many objects of the same type, and you want to avoid having to create them from scratch each time.

* **Singleton:**

Ensures that a class has only one instance and provides a global point of access to it. This pattern is useful when you need to ensure that there is only one instance of a particular class, such as a database connection or a logger.