There are three basic classifications of patterns Creational, Structural, and Behavioral patterns.**=BSC**

**Creational Patterns**

• **Abstract Factory**:- Creates an instance of several families of classes  
• **Builder**: - Separates object construction from its representation  
• **Factory Method**:- Creates an instance of several derived classes  
• **Prototype**:- A fully initialized instance to be copied or cloned  
• **Singleton**:- **A class in which only a single instance can exist**

**Note**: - The best way to remember Creational pattern is by remembering ABFPS (Abraham Became First President of States).  
Structural Patterns

• **Adapter**:-Match interfaces of different classes .  
• **Bridge**:-Separates an object’s abstraction from its implementation.  
• **Composite**:-A tree structure of simple and composite objects.  
• **Decorator**:-Add responsibilities to objects dynamically.  
• **Façade**:-**A single class that represents an entire subsystem.**  
• **Flyweight**:-A fine-grained instance used for efficient sharing.  
• **Proxy**:-An object representing another object.

• **Mediator**:-Defines simplified communication between classes.  
• **Memento**:-Capture and restore an object's internal state.  
• **Interpreter**:- A way to include language elements in a program.  
• **Iterator**:-**Sequentially access the elements of a collection.**  
• **Chain of Resp**: - **A way of passing a request between a chain of objects.**  
• **Command**:-Encapsulate a command request as an object.  
• **State**:-Alter an object's behavior when its state changes.  
• **Strategy**:-Encapsulates an algorithm inside a class.  
• **Observer**: - A way of notifying change to a number of classes.  
• **Template Method**:-Defer the exact steps of an algorithm to a subclass.  
• **Visitor**:-Defines a new operation to a class without change.

<https://www.codeproject.com/Articles/28309/Design-pattern-FAQ-Part-Training>

MCQ: <https://www.sanfoundry.com/advanced-java-questions-answers-design-patterns/>