1) Creator Pattern

Who should be responsible for creating a new instance of some class?

-Assign the responsibility to a class that contains or aggregates the created object.

-Assign responsibility to a class that uses the created object

-Assign responsibility to a class that records the created object

Creator pattern connects creator to the created object which supports low coupling but initializing complex data can be problem for creator.

Pros:

Choosing the correct creator supports low coupling and cohesion.

Cons:

Assigning many responsibilities to a single class can negatively affect modularity.

2) Information Expert Pattern

Who should be responsible for this objective?

- Assign responsibilities to classes with partial information

- Assign responsibility to the class with all information

- Assign responsibility to a new class that gathers information.

Assigning all responsibilities keep setup cohesive but may cause bloated classes while distributed classes increase modularity but also increase coupling.

Pros:

More efficient as there is main information expert

Cons:

Can result with large complexity

Can cause large classes

3) Controller Pattern

What first object beyond the UI layer receives and coordinates ("controls") a system operation?

- Use use case controllers

- Use a facade controller

Facade controllers can be bloated. Use case controllers support modularity but they are harder to manage.

Pros:

Facade controllers are easier to coordinate while use case controllers provide modularity.

Cons:

Use case contollers are harder to manage and facade controllers can be bloated