1. Creator: Quem cria uma instância de A?

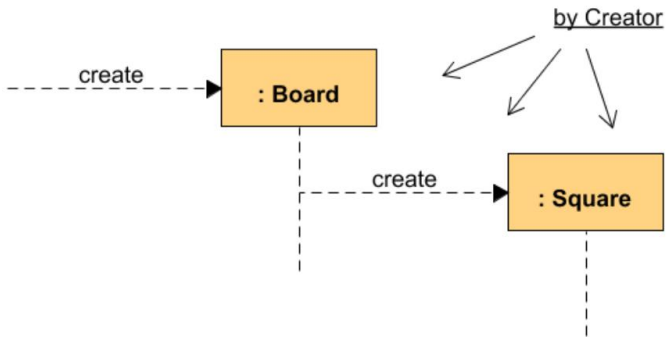
Solução: Assign a uma classe B a responsabilidade de criar instâncias da classe A se:

– B contains or aggregates A (in a collection)

– B records A

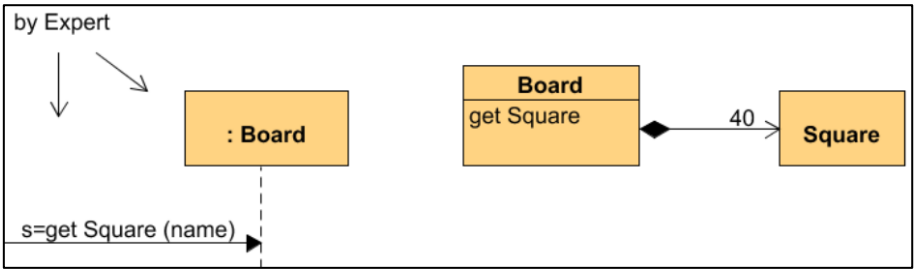
– B closely uses A

– B has the initializing data for A



2. Information Expert: Como atribuir responsabilidades aos objects?

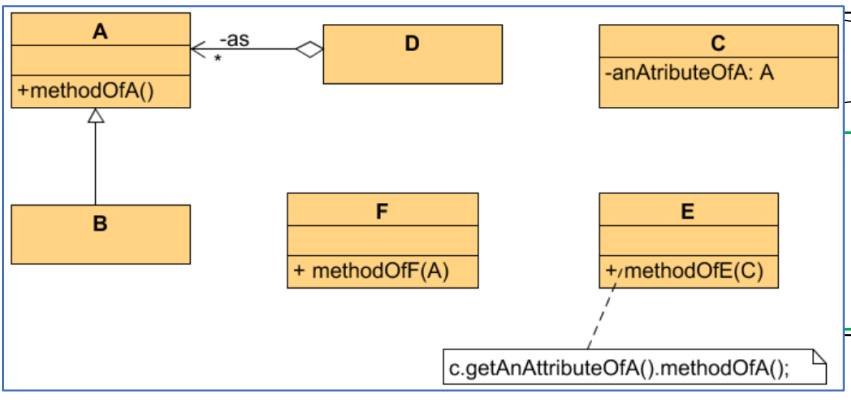
Assign responsibility to the class that has the information needed to fulfill it

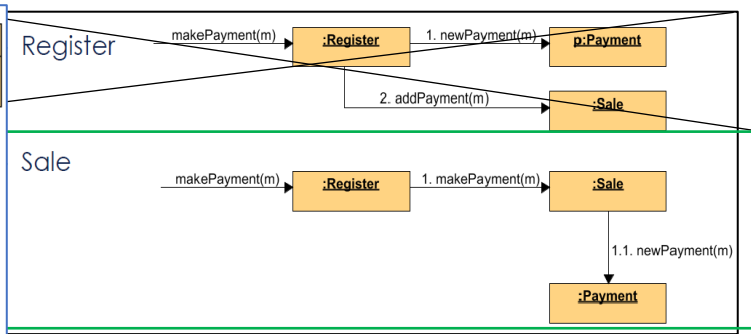


3. Low Coupling: Como reduzir o impacto de mudanças e encorajar ao reuse?

Assign a responsibility so that coupling (linking classes) remains low.

Try to avoid one class to have to know about many others.

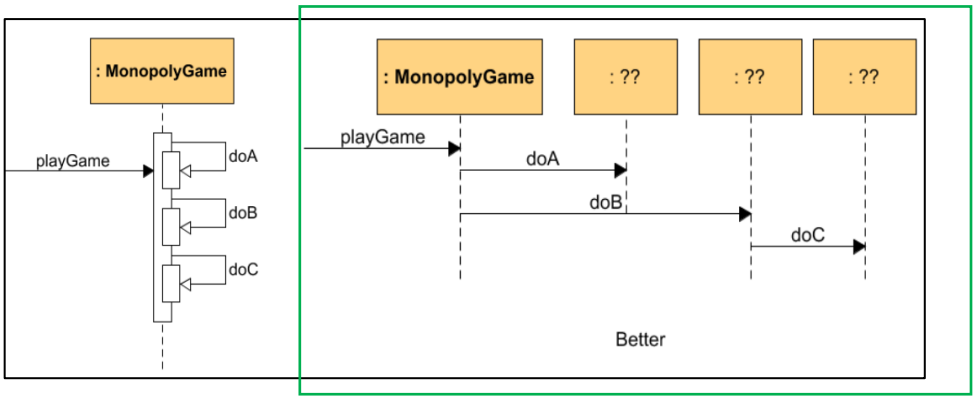




4. High Coesion: Como manter as classes focadas e manageable?

Assign responsibility so that cohesion remains high. Delegate responsibility & coordinate work

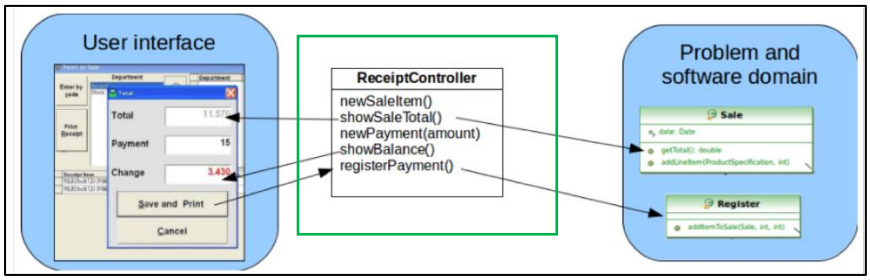
Complements Low Coupling



5. Controller: Quem deve ser responsável pelos UI events?

Non-user interface responsável por lidar e receber system events.

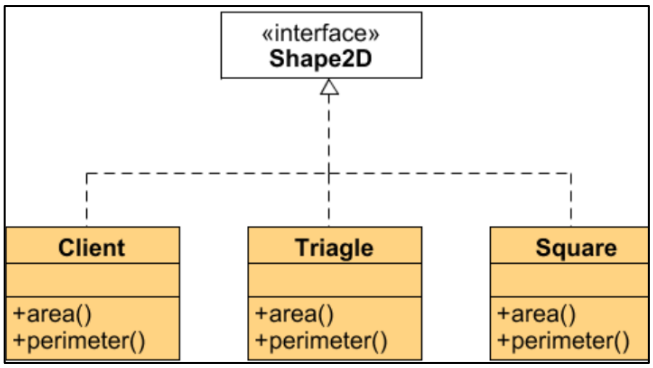
Assign responsabilidade a esta interface.



6. Polymorphism: Como lidar com o comportamento com base no tipo sem if/esle ou switch?

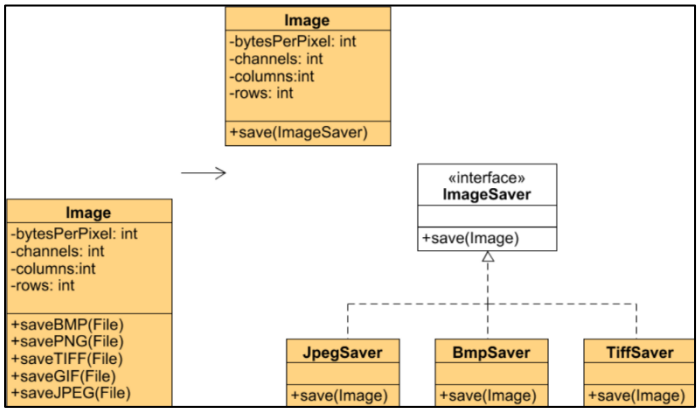
Polymorphic methods: giving the same name to (different) services in different classes. ervices are implemented by methods.

Polimorfismo: Dar o msesmo nome a métodos de diferentes classes.



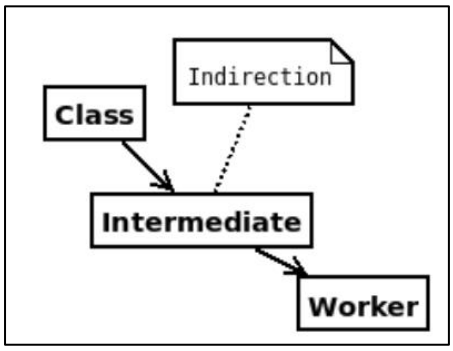
7. **Pure Fabrication: What object should have a responsibility when no class of the problem domain may take it without violating High Cohesion and Low Coupling?**

Atribuir responsabilidades de alta coesão a uma classe artificial que não representa nada no domínio do problema.



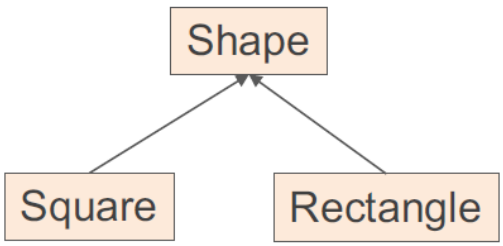
8. **Indirection: Como evitar Direct Coupling?**

Atribuir responsabilidade a um objeto intermediário para mediar entre outros componentes, não ficando diretamente acoplados.



9. **Protected Variations: How to design objects, subsystems, and systems so that variations or instabilities in the elements do not have na undesirable impact on other elements?**

Identificar pontos de variação ou de instabilidade previstos e atribuir responsabilidades para criar uma interface estável em torno deles.



- **Liskov Substitution Principle (LSP)**

A subclass B of A should be substitutable for superclass A, i.e., B should be a true subtype of A

---Um quadrado não é um true subtype de retângulo.

- **Law of Demeter (Don't talk to strangers): Como evitar saber a estrutura de objetos indiretos?**

Se 2 classes não têm razões para estarem diretamente informados uma da outra, ou de outra maneira acoplados, então não devem interagir diretamente.