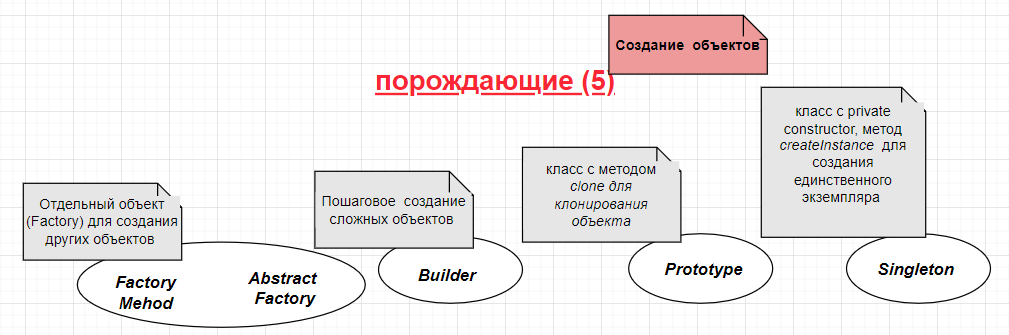
**Паттерны проектирования**

**ИСиТ,ПОИБМС-2**

**Лекция 11**

**Итоги**

**Всего 22 паттерна**

***Factory Method***

1. *IProduct, ConcreteProduct\_:IProduct*
2. *ICreator, ICreator.createProduct():IProduct*
3. *ConcreteCreator\_:ICreator, ConcreteCreator\_.createProduct():IProduct*

***Abstract Factory***

1. *IProduct\_,…*
2. *ConcreteProduct\_:IProduct\_,…*
3. *IAbstractFactory, IAbstractFactory.CreateProduct\_:IProduct\_,…*
4. *ConcreteFactory\_:IAbstractFactory*

***Builder***

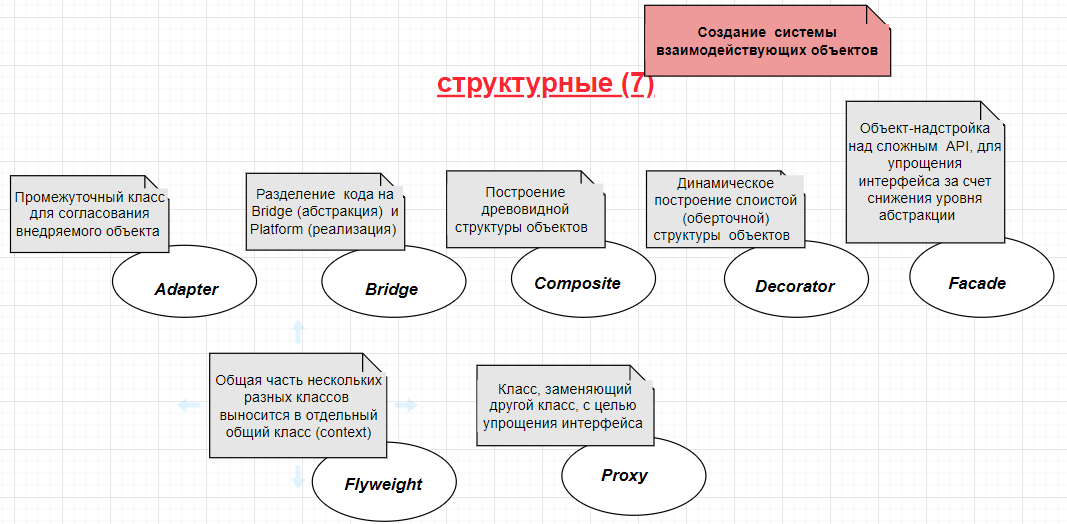
1. *Product\_,…*
2. *IBuilder, IBuilder.step\_(),…*
3. *ConcreteBuilder\_:IBuilder,…, ConcreteBuilder\_,getResult():Product\_*
4. *Director, new Director.Director(typeBuild), Director.builder:IBuilder*

***Prototype***

1. *IProduct, IProduct.clone(id):IProduct*
2. *Product\_:IProduct…, Product\_.clone(id):IProduct*

***Singleton***

1. *Singleton, private Singleton.Singleton(), static Singleton.createInstance(): Singleton.*

****

***Adapter***

1. *IExternal, IExtenal.met(X):Y*
2. *Client, new Client.Client(IExternal)*
3. *S, S.metS(Z):W*
4. *Adapter:IExternal, Adapter.met(X){W =S.metS; …,return Y}*
5. *New Client(Adapter)*

***Bridge***

1. Bridge =  *IProperty, IAbsraction, IAbstraction.IProperty*
2. Platform = *Implementation:IPoperty, Realisation:IAbstraction*,
3. *Client, new Client(IAbstraction)*

***Composite***

1. *IComponent, IComponent.execute()*
2. *Composite:IComposite, Composite.children:IComposite[], Composite.add(IComponent), Composite.del(IComponent), getChildren():IComponent[].*

***Decorator***

1. *IComponent, IComponent.execute()*
2. *Component\_:IComponent*
3. *Decorator:IComponent, Decorator.wrapped, new Decarator.Decarator(IComponent)*
4. *ConreteDecorator\_:Decorator*

***Facade***

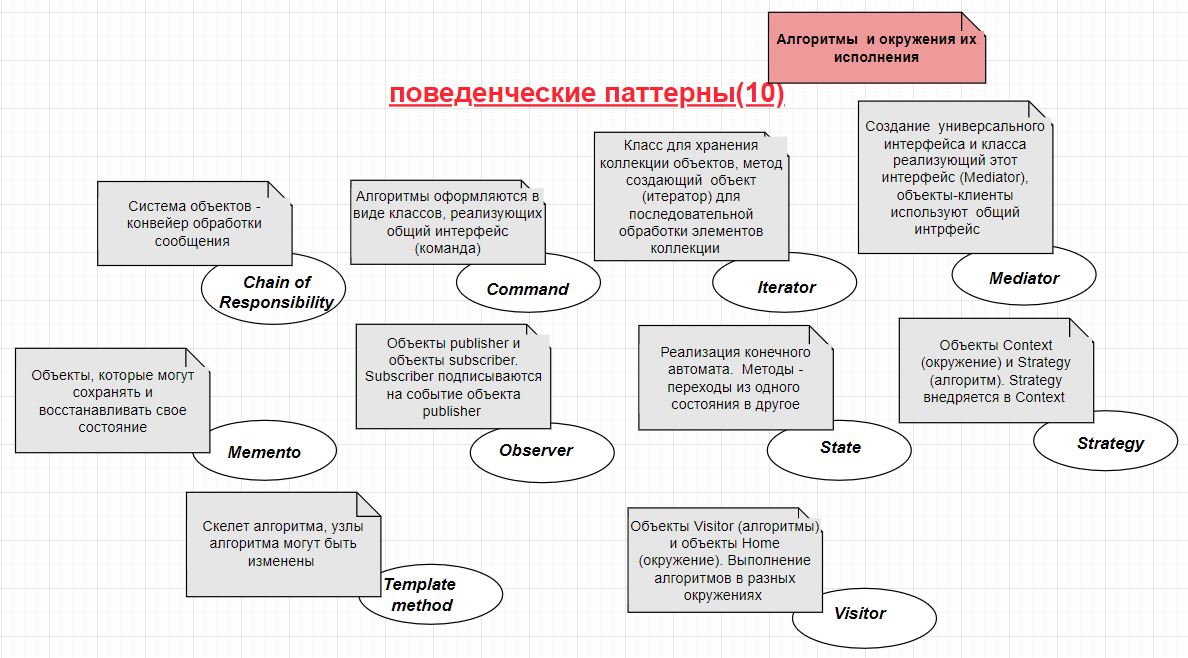
1. *API*
2. *SubAPI\_,….*

***Flyweight***

1. *CommonAPI*
2. *FlyweightAPI\_,…, FlyweightAPI\_.FlyweightAPI\_(CommonAPI)*

***Proxy***

1. *ServerAPI*
2. *ServerStub*
3. *ServerProxy*
4. *Client*

****

***Chain of Responsibility***

1. *ILink, ILink.get(message):message, ILink.put(message):message*
2. *Link\_:ILink, ..., Link\_.get(message),Link\_.put(message)*
3. *Chain, Chain.add(ILink)*
4. *Recipient, …, Recipient.Recipient(Chain), Recipient.get(message)*
5. *Sender\_:, …,*

*Sender\_.put(recipient, message){ recipient.get(message) } .*

***Command***

1. *ICommand, ICommand.met\_()*
2. *AbstractInvoker, AbstractInvoker.AbstractInvoker(ICommand), AbstractInvoker.command:ICommand, AbstractInvoker.invoc()*
3. *Invoker\_: AbstractInvoker*

***Itterator***

1. *IIterator*
2. *ICollection, ICollection.createItterator():IIterator*
3. *Iterator\_:IIterator*
4. *Collection\_:ICollection, Collection\_.createIterator():Iterator\_*

***Mediator***

1. *IMedeiator, IMediator.notify(componetname)*
2. *Component\_,Componet\_.Component\_(IMediator)*
3. *Mediator:Imediator, Mediator.component\_*

***Memento***

1. *Memento, Memento.Memento(state)*
2. *Caretaker, Caretaker.history:Memento[], Caretaker.push(Memento), Caretaker.pop():Memento*
3. *Originator, Originator.save(), Originator.restore().*

***Observer***

1. *ISubscriber, ISubscriber.update()*
2. *Subscriber\_:ISubsriber*
3. *Publisher, Publisher.subcribers: ISubscriber, Publisher.subscibe(IPublisher), Publisher.unsubscibe(IPublisher)*

***State***

1. *State, State.name*
2. *Machine, Machine.state:State, Machine.changeState(State)*

***Strategy***

1. *IStrategy, IStrategy.execute()*
2. *Context, Context(IStrategy), Context.Context(IStrategy), Context.set(IStrategy)*
3. *Strategy\_:IStrategy, Strategy.Strategy(Context)*

***Template method***

1. *AbstractProcess, virtual AbstractProcess.start(), virtual AbstractProcess.step\_()*
2. *Process\_: AbstractProcess*

***Visitor***

1. *IHome, IHome.accept(IVisitor)*
2. *Home\_: IHome*
3. *IVisitor, IVisitor.visit(Home\_)*
4. **КОНЕЦ**