Proposta para a aula:

Construir um simulador para um jogo como exemplo de projeto de desenvolvimento. Estudar implementações. Usar isso para recordar e apresentar conceitos de programação orientada a objetos.

Etapas do projeto:

- Especificação do processo a simular;
- Listagem dos requisitos do software;
- Especificação do software
  - O que deve ser representado no programa
  - Como deve ser representado no programa
- Implementação;
- Teste.

Nota: neste nível, não existe metodologia única - as tarefas podem ter outros nomes e outros objetivos. Também é comum que se especifique o que representar tendo em vista como representar, desta forma, quem sabe fazer uma boa especificação **TEM QUE SABER PROGRAMAR**, e de preferência ser um bom programador, para prever e evitar problemas que atrasem o desenvolvimento, como veremos à frente.

Etapa 1 - Especificação do processo

Deseja-se simular o seg. jogo de cartas:

Cartas têm números de 1 a 13 e naipes Paus, Copas, Espadas e Ouros, inicialmente o monte tem 52 cartas.

Os jogadores se apresentam para jogar.

O jogo tem uma única rodada nela cada jogador recebe uma carta. Ao final todos mostram a carta que receberam e vence o que tiver a carta de maior número. Caso haja empate, vence a de naipe mais alto.

Etapa 2 - Requisitos

Um número arbitrário de jogadores pode jogar.

O programa resultante tem que ser uma aplicação Java.

...

Etapa 3 - Especificação do software.

Nota: já analisando o processo em termos de orientação a objetos!!!!

Quais são os objetos?

Jogadores
Cartas

O que é importante na carta e o que a carta faz??

Pois é, depende da carta, e dependendo do jogo, atributos como cor ou naipe não são importantes.

Em geral uma carta não é sujeito da ação. Ela é sujeita a uma ação. (uma carta não se vira - o jogador a vira). Pensando na programação, vale a pena reforçar que como o programador pode escolher (quase) qualquer nome e comportamento para os métodos, ele pode criar um método vira na classe carta. A invocação (chamada) fica algo como **carta.vira (...).** Soa estranho, mas às vezes é conveniente dar nomes assim. Isso não quer dizer que a carta se vire. Ou seja, fazer os

nomes aderentes ao comportamento é uma decisão do programador.

E o que é importante no jogador e o que ele faz?

Os jogadores jogam - armazenam as suas cartas, compram, descartam, usam estratégias,...

...além de jogar, coletivamente, cuidam que as regras sejam cumpridas. Em geral se alguém infringe alguma regra, os outros se manifestam e há um consenso sobre a aplicação da regra. Por exemplo, sempre tem o que esquece de jogar e o que pula a vez do outro. Como se resolve isso numa partida real? Programar esse comportamento exige que todos os jogadores verifiquem se as regras são cumpridas e requer que haja como comunicar infrações e que se chegue a um consenso. Programar isso pode ser mais difícil que programar o jogo... há como evitar esse trabalho?

Nota: só pensa nisso quem é observador, sabe (o que é difícil) programar e conhece ou saca como resolver.

Sim! criando um novo objeto que garanta que as regras sejam cumpridas, um espécie de juiz ou administrador. O que deve fazer o administrador?

administrar a entrada/saída de jogadores, cuidar da distribuição das cartas, caso haja apostas, cuidar do "dinheiro",... esse administrador poderia inclusive saber que cartas foram dadas a cada jogador e conferir se houve trapaça a cada jogada, ou até armazenar as cartas de cada jogador, mostrando-as para o jogador certo. Enfim, isso é para quando formos implementar.

Ficamos então com jogadores, cartas e administrador... mais algum objeto??
Vamos detalhar esses objetos:

Carta, nesse caso tem número e naipe. Algum desses pode ser um objeto??
As cartas são usadas sozinhas?
Como o administrador controla a ordem de jogada??

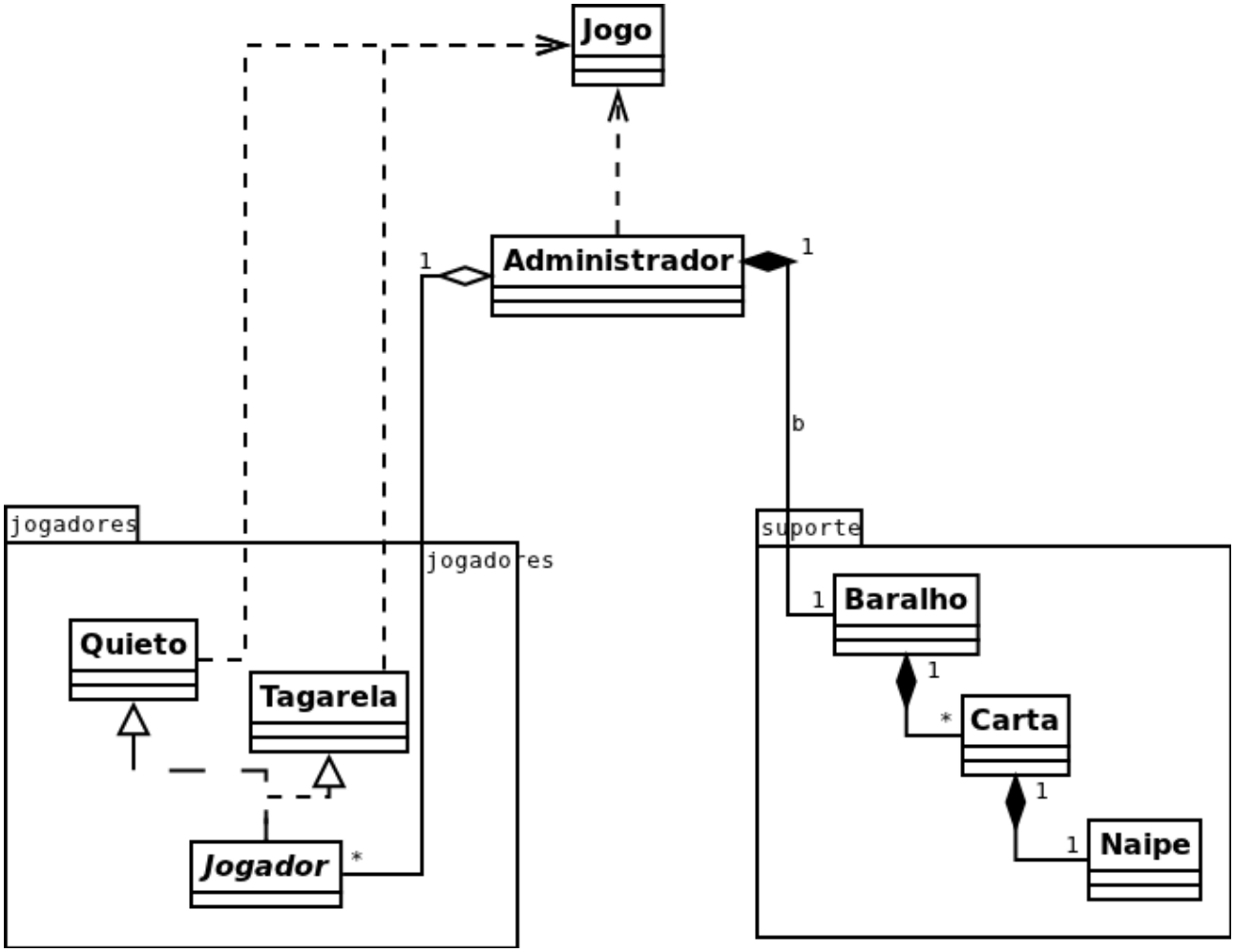
Depois de satisfeitos com os objetos. Qual é o fluxo PRINCIPAL do processo/programa??

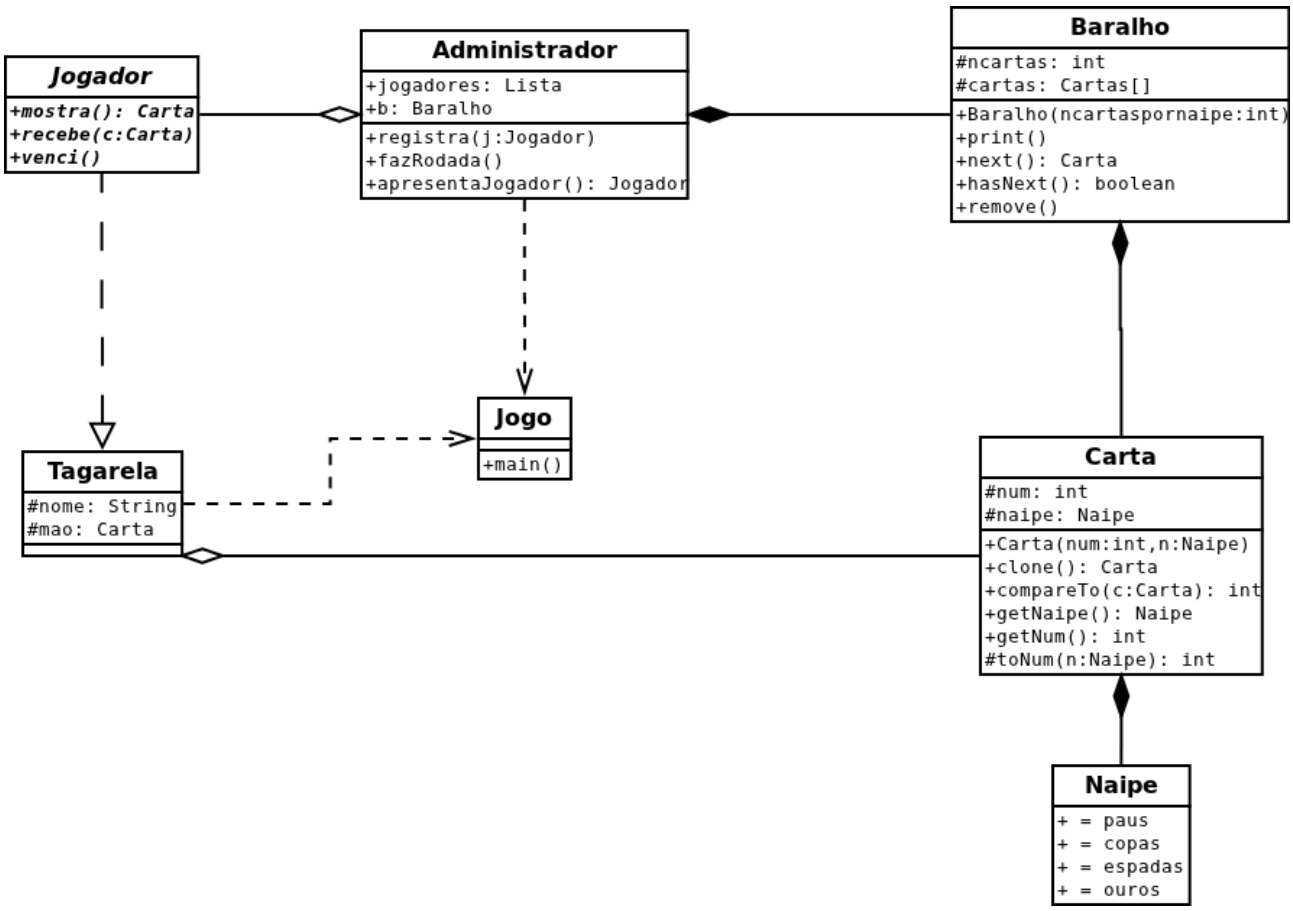
Nota: começar pelo mais amplo e ir para o mais detalhado é o que se chama metodologia *top-down*
ir do mais detalhado para o mais amplo é o que se chama metodologia *bottom-up*
como você já deve ter notado, na prática usa-se um pouco de uma e um pouco de outra. É muito difícil um projeto em que se use apenas uma.

```
main {
    instancia administrador
    instancia jogadores
    administrador registra jogadores
    faz uma rodada
    apresenta o vencedor.
}
```

Note que as ações ligadas ao jogo propriamente dito são todas do administrador, e de fato, se estivermos interessados apenas no resultado, esse jogo é equivalente a sortear uns tantos números (sem reposição), vence quem tiver o maior número. (Código no JogoCompleto.java).

... entretanto aproveitar esse código e construir uma interface gráfica, ou criar jogos com regras mais sofisticadas ou jogos com conexão remota exige que o código seja totalmente refeito. É aí que entra a orientação a objetos.





Tarefas para o laboratório amanhã

1-) Experimentem trocar especificadores de acesso (ou deixar em branco) e ponham tudo para funcionar (especificadores de acesso em ação).
2-) Experimentem mudar os arquivos de pasta e ponham tudo para funcionar (pacotes e import).
3-) Construam um jogador trapaceiro.
4-) Ajustem o administrador para que o trapaceiro não funcione mais. É fácil fazer o administrador trapacear??
5-) Modifiquem o jogo livremente, por exemplo, permitindo apostas, trocas de carta, transformando-o em um vinte e um....