DIM00506
Projeto Detalhado de Software

Professor: Uirá Kulesza

DIMAp / UFRN, 2012.2

Aula 3:
Princípio Open-Closed

Princípio 2
The Open-Closed

“All systems **change** during their life cycles. This **must be born in mind** when developing systems are expected to **last longer** than the first version”

Ivar Jacobson
(Ericsson, Rational)

Contribuições: Metodologia OO, Use Cases, UML

Como criar projetos que são estáveis frente a mudanças/ manutenções futuras?

Princípio Open-Closed

"Entidades de Software (classes, módulos, funções, etc) deveriam ser abertas para extensão, mas fechadas para modificação" [Meyer 97]

Traduzindo...

- Módulos
  - Abertos para Extensão
    - O comportamento do módulo pode ser estendido, a medida que novas demandas surgem
  - Fechado para Modificação
    - Estender o comportamento do módulo não resulta em mudanças no código fonte ou binário do módulo

Mas como conseguimos estender o comportamento de módulos sem modificá-los?

Abstração

- As linguagens OO permitem definir abstrações fixas, mas que ainda não tem seu comportamento definido

- Exemplo: Classes Abstratas, Interfaces

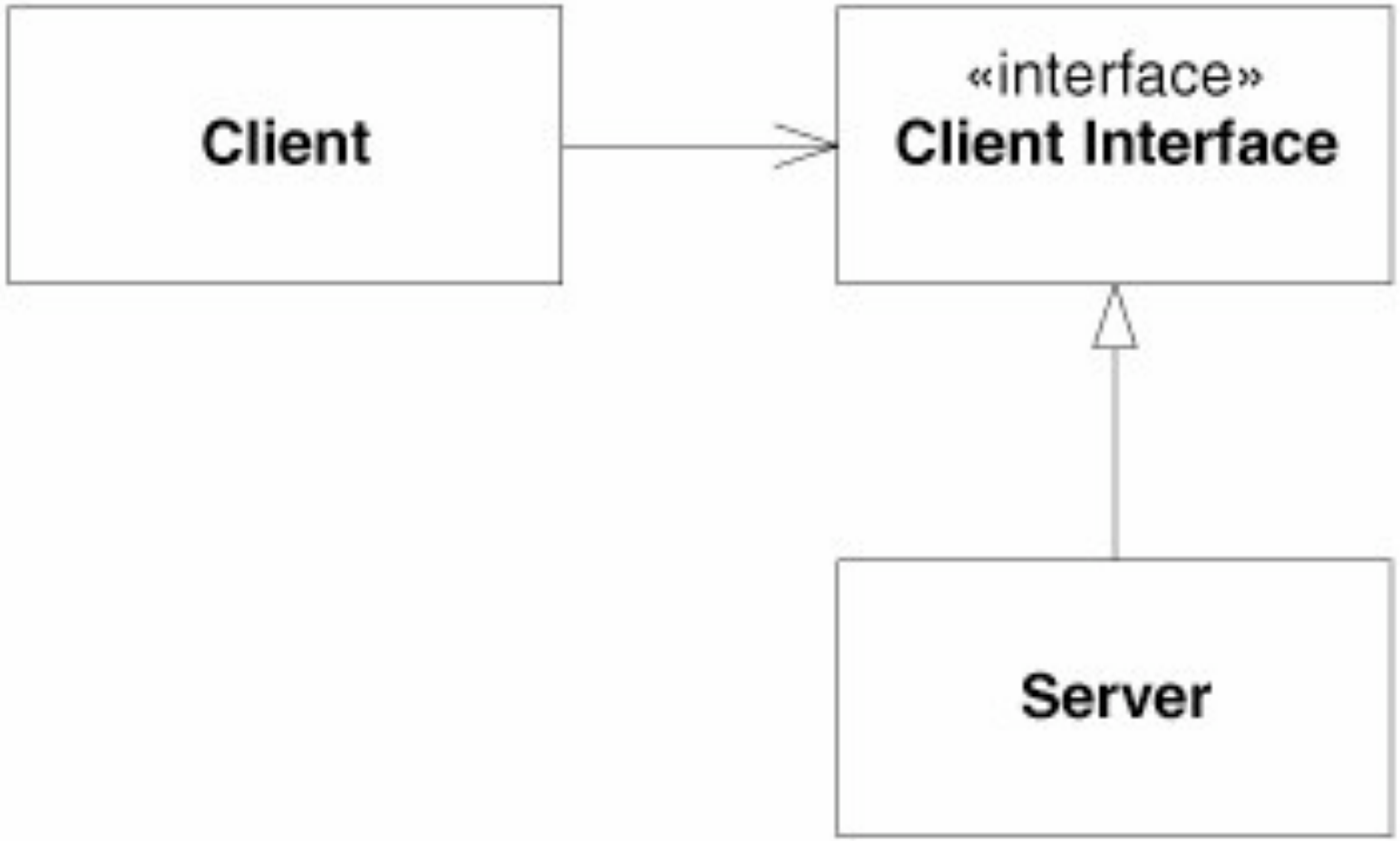
- Abstraem comportamento que pode ser estendido

Projeto Open-Closed?

Como tornar tal projeto Open-Closed?

- Ambas as classes são concretas
- O cliente usa uma classe Server, se precisarmos que ele use uma classe diferente temos que mudar isso internamente no código dele

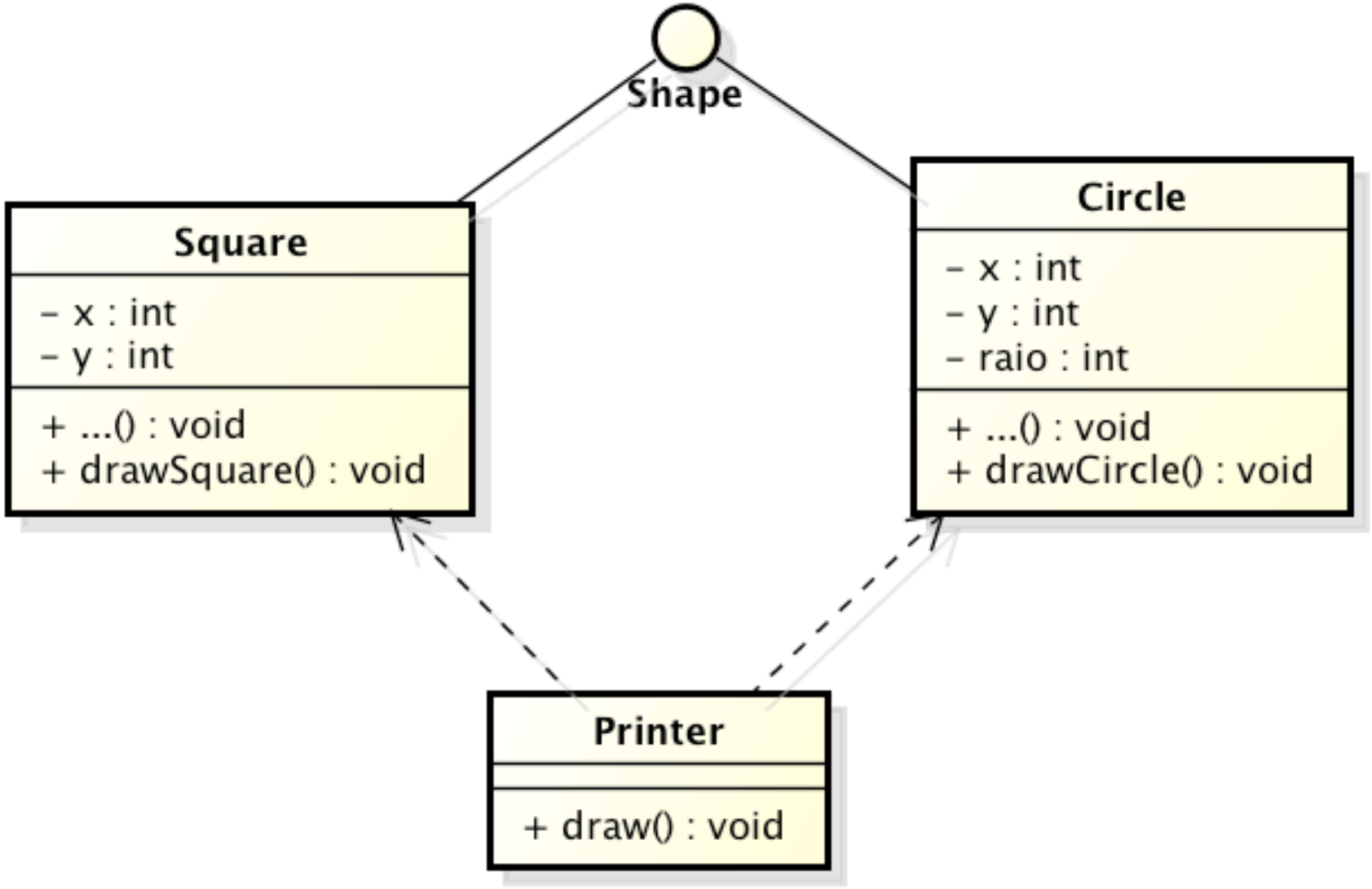
Projeto Open-Closed



Aplicação Shape

- Considere uma aplicação Shape para pintar círculos e quadrados em uma GUI
- Eles devem ser pintados em uma determinada ordem
  - Círculos antes de quadrados
- A aplicação deve navegar na lista de figuras geométricas e imprimí-las

Aplicação Shape



Código de Printer

```
public class Printer2 {

   public void drawAllShapes(List<Shape> shapes){

      for (Shape element: shapes){

         if (element instanceof Square){

            Square square = (Square) element;
            square.drawSquare();

         }else if (element instanceof Circle){

            ((Circle) element).drawCircle();
         }
      }// for

   }// method

}// class
```

Tal aplicação segue o princípio open-closed?

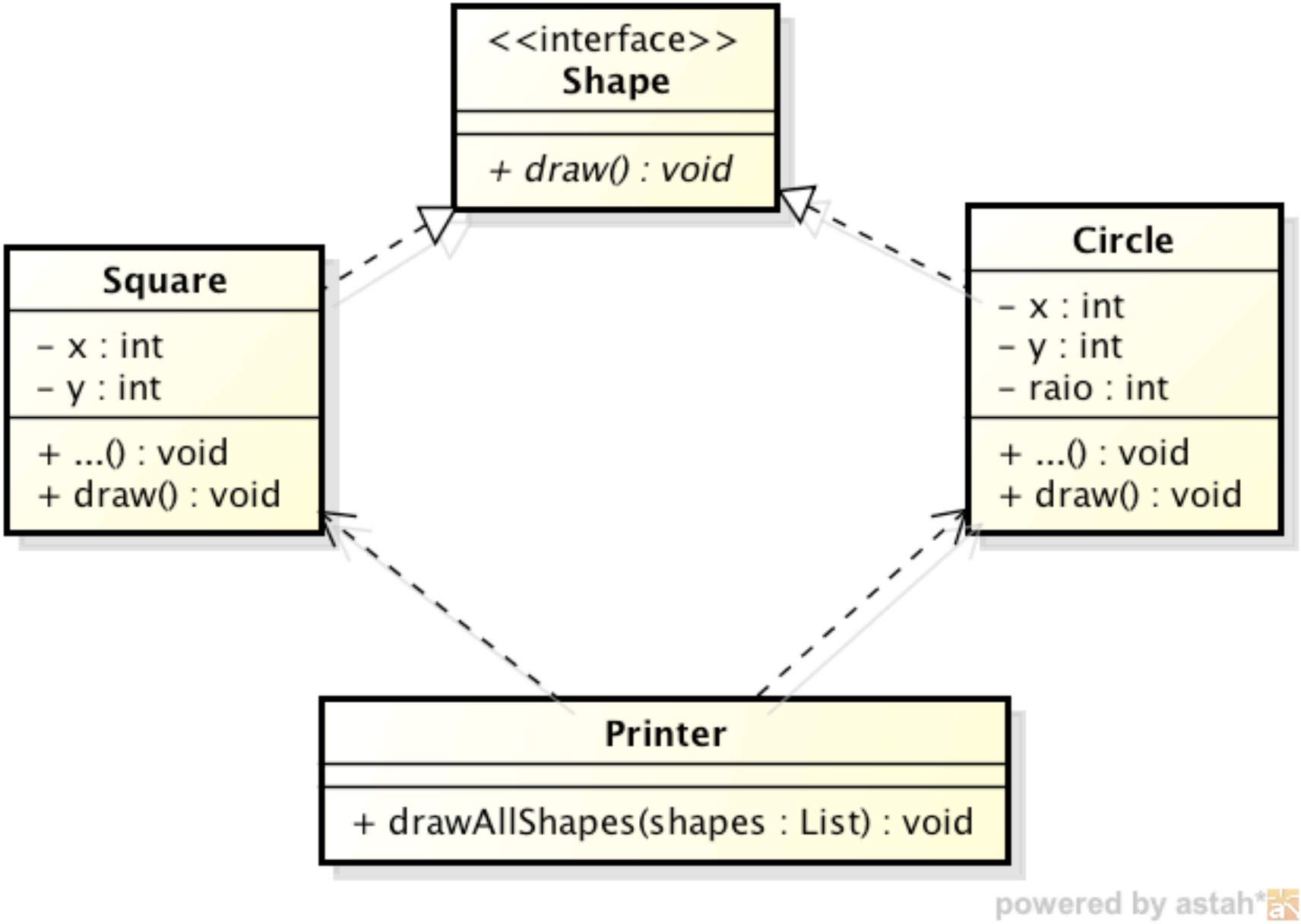
O que acontece com o projeto se quisermos adicionar uma nova classe triângulo?

Conseqüências

- Rigidez
  - Classe Printer precisa ser recompilada e redistribuída
  - Ao levar classe Printer para um projeto é necessário levar todas as outras classes junto
- Difícil de entender e manter
  - Dificuldade de encontrar e entender os diferentes switch/case ou if/else

Que tal pensarmos em projeto open-closed para a aplicação Shape?

Aplicação Shape – Open/Closed



Código de Printer

```
public class Printer {

   public void drawAllShapes(List<Shape> shapes){

      for (Shape element: shapes){
         element.draw();
      }

   }
}
```

Conseqüências

- Inclusão de nova figura geométrica (Triângulo) não afeta o comportamento do método drawAllShapes()
- Adição da classe Triangulo não afeta outras classes envolvidas no projeto
  - Não há necessidade de recompilar ou reinstalar classes
- O projeto permite mudar a aplicação pela adição de novas classes, ao invés de ter que mudar o código de classes existentes…

Faltou algum requisito?

Como imprimir todos os círculos antes dos quadrados?

Antecipação de Mudanças

- O projeto não está preparado para tal mudança
- Não existe projeto que esteja preparado para diferentes contextos
- Projetista deve elaborar estruturas responsáveis por atender determinadas mudanças
  - Probabilidade de determinadas mudancas ocorrer

O que podemos fazer para deixar nosso projeto preparado para mudanças?

Antecipação de Mudanças

- Entender bem o domínio de aplicação
  - identificar seus requisitos mais voláteis e prioritários
- Usar experiência e bom senso baseado em projetos anteriores

Antecipação de Mudanças

- Fazer um projeto flexível considerando as possíveis mudanças
  - Evitar complexidade desnecessária
- Esperar mudanças futuras ocorrerem para avaliar impacto no projeto...

Mas...

E o projeto para imprimir círculos antes dos quadrados?

Ficará como desafio para próxima aula...

Antecipação de Mudanças

- Fazer um projeto flexível considerando as possíveis mudanças
  - Evitar complexidade desnecessária

- Esperar mudanças futuras ocorrerem para avaliar impacto no projeto...

Considerações Finais

- O princípio open-closed está no "coração" do projeto OO
- Seguindo tal princípio, vários benefícios de OO podem ser alcançados:
  - Flexibilidade, reusabilidade, manutenibilidade
- Simples uso de linguagens OO não garante tal princípio
- Requer sabedoria do projetista para aplicar flexibilidade apenas para porções do projeto que mudam frequentemente

Tarefas

- Decidir quem será sua dupla no projeto e qual projeto irão refatorar/melhorar na 1ª etapa do projeto

- Resolver o exercício da ordenação de imprimir círculos antes dos quadrados na aula !

Perguntas / Questões / Dúvidas ?

Referências

- R. Martin, Agile Software Development: Principles, Patterns and Practices, Prentice Hall, 2002.