Projeto orientado a objetos

- Projetar sistemas usando objetos auto-contidos e classes de objetos

Objetivos

- Explicar como um projeto de software pode ser representado como um conjunto de objetos que interagem e gerenciam o seu próprio estado e operações
- Descrever atividades no processo de projeto orientado a objetos
- Apresentar diversos modelos que descrevem um projeto orientado a objetos
- Mostrar como a UML pode ser usada para representar estes modelos

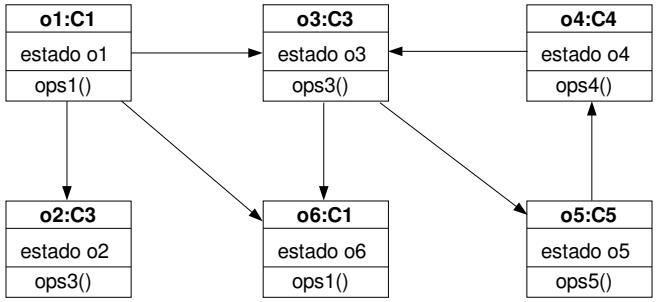
Tópicos abordados

- Objetos e classes de objetos
- Um processo de de projeto orientado a objetos
- Evolução de projeto

Características do POO

- Objetos são abstrações de entidades do mundo real ou do sistema e são auto-gerenciáveis
- Objetos são independentes e encapsulam estado e a representação das informações.
- A funcionalidade do sistema é expressa em termos de serviços de objetos
- Áreas de dados compartilhadas são eliminadas. Os objetos se comunicam por envio de mensagens
- Os objetos podem ser distribuídos e podem executar seqüencialmente ou em paralelo

Objetos interagindo



Vantagens do POO

- Manutenção mais fácil. Os objetos podem ser compreendidos como entidades *stand-alone*
- Os objetos podem ser vistos como componentes reutilizáveis
- Para alguns sistemas, existe um mapeamento óbvio das entidades do mundo real para objetos do sistema

Desenvolvimento orientado a objetos

- Análise, projeto e programação orientados a objetos são atividades relacionadas, mas distintas
- A *Análise Orientada a Objetos* dedica-se a desenvolver um modelo de objetos do domínio da aplicação
- O *Projeto Orientado a Objetos* preocupa-se em desenvolver um sistema orientado a objetos para implementar requisitos
- A *Programação Orientada a Objetos* preocupa-se em realizar um projeto de software usando uma linguagem de programação OO, como Java ou C++

Objetos e classes de objetos

- Objetos são entidades em um sistema de software que representam instâncias do mundo real e entidades do sistema
- Classes de objetos são modelos para objetos. Elas podem ser usadas para criar objetos
- Classes de objetos podem herdar atributos e serviços de outras classes de objetos

Objetos e classes de objetos

- Um objeto é uma entidade que possui um estado e um conjunto definido de operações que operam nesse estado.
- O estado é representado por um conjunto de atributos de objeto.
- As operações associadas com o objeto fornecem serviços para outros objetos (clientes), que requisitam esses serviços quando alguma computação é necessária.

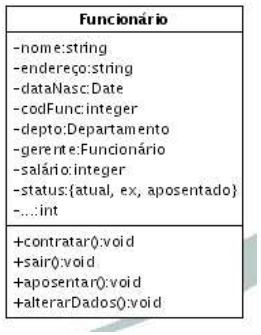
Objetos e classes de objetos

- Os objetos são criados de acordo com uma definição de classe de objetos, que serve como um modelo (*template*) para criar objetos.
- A classe apresenta declarações de todos os atributos e operações que devem ser associados a um objeto dessa classe.

Unified Modeling Language (UML)

- Várias notações diverentes para descrever projetos orientados a objetos foram propostas nas décadas de 1980 e 1990
- A *Unified Modeling Language* é uma integração dessas notações
- Ela descreve notações para uma gama de modelos diferentes que podem ser produzidos furante a análise e projeto OO
- Ela é agora um padrão *de facto* para modelagem OO

Classe de objeto Empregado (UML)



Comunicação entre objetos

- Conceitualmente, objetos se comunicam por envio de mensagens.
- Mensagens
  - O nome do serviço requerido pelo objeto chamador.
  - Cópias da informação requerida para executar o serviço e o nome de um responsável pelo resultado do serviço.
- Na prática, as mensagens são freqüentemente implementadas por chamadas de procedimentos
  - Nome = nome do procedimento.
  - Informação = lista de parâmetros.

Exemplos de mensagens

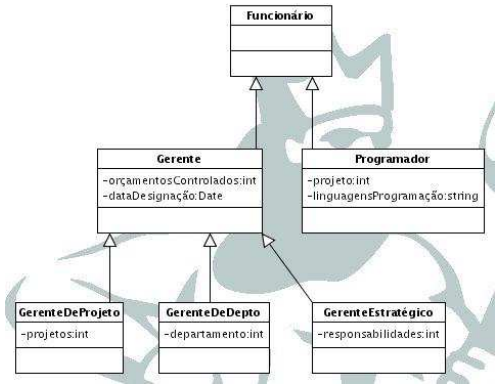
```
// Chama um método associado com um
// objeto buffer que retorna o próximo valor
// no buffer
        v = bufferCircular.Get () ;

// Chama o método associado com um
// objeto termostato, que define a
// temperatura a ser mantida
        termostato.setTemp (20) ;
```


Generalização e herança

- Objetos são membros de classes que definem os tipos dos atributos e das operações
- As classes podem ser organizadas em uma hierarquia de classes onde uma classe (*super-classe*) é uma generalização de uma ou mais outras classes (*sub-classes*)
- Uma sub-classe herda os atributos e operações da sua super-classe e pode incluir novos métodos e atributos próprios
- A generalização na UML é implementada como herança nas LPOO

Uma hierarquia de generalização



Vantagens da herança

- É um mecanismo de abstração que pode ser usado para classificar entidades
- É um mecanismo de reuso tanto em nível de projeto quanto de programação
- O grafo de herança é uma fonte de conhecimento organizacional sobre domínios e sistemas

Problemas com a herança

- Classes de objetos não são auto-contidas. Elas não podem ser compreendidas sem referências a sua super-classe
- Os projetistas têm uma tendência a reusar o grafo de herança criado durante a análise. Isso pode levar a uma ineficiência significante
- Os grafos de herança da análise, projeto e implementação possuem funções diferentes de devem ser mantidos separadamente

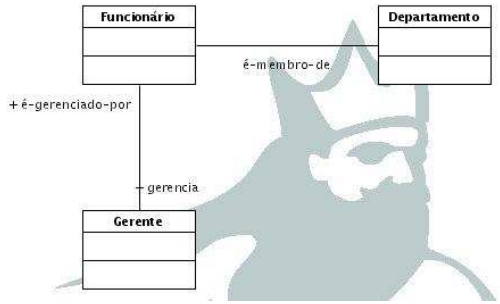
Herança e o Projeto Orientado a Objetos

- Existem visões diferentes de quanto fundamental a herança é para o Projeto OO.
  - Visão 1. Identificar a hierarquia ou rede de herança é uma parte fundamental do projeto orientado a objetos. Obviamente, isso pode apenas ser implementado usando uma LPOO.
  - Visão 2. A herança é um conceito útil de implementação, que permite o reuso de definições de atributos e operações. Identificar uma hierarquie de herança no estágio de projeto impõe restrições desnecessárias à implementação
- A herança introduz complexidade, e isso é indesejável, principalmente em sistemas críticos

Associações em UML

- Objetos e classes participam em relacionamentos com outros objetos e classes
- Em UML, um relacionamento genérico é indicado por uma associação
- As associações podem ser anotadas com informações que descrevem a associação
- As associações são gerais, mas podem indicar que um atributo de um objeto é um objeto associado ou que um método depende de um objeto associado

Um modelo de associação



Objetos concorrentes

- A natureza dos objetos como entidades auto-contidas faz com que eles sejam adequados para implementação concorrente
- O modelo de envio de mensagens da comunicação de objetos pode ser implementado diretamente se os objetos estiverem rodando em processadores separados em um sistema distribuído

Servidores e objetos ativos

- Servidores.
  - O objeto é implementado como um processo paralelo (servidor), com pontos de entrada correspondentes às operações do objeto. Se não forem feitas chamadas a ele, o objeto suspende a si mesmo e aguarda por requisições posteriores de serviço
- Objetos ativos
  - Os objetos são implementados como processos paralelos e o estado interno do objeto pode ser alterado pelo próprio objeto ou por chamadas externas

Objeto *transponder* ativo

- Objetos ativos podem ter seus atributos modificados por operações, mas podem também atualizá-los autonomamente, usando operaçoes internas
- O objeto *transponder* envia a posição da aeronave. A posição pode ser atualizada usando um sistema de posicionamento via satélite. O objeto periodicamente atualiza a posição pela triangulação dos satélites

Um objeto *transponder* ativo

```
class Transponder extends Thread {

        Position currentPosition ;
        Coords c1, c2 ;
        Satellite sat1, sat2 ;
        Navigator theNavigator ;

        public Position givePosition ()
        {
                return currentPosition ;
        }

        public void run ()
        {
                while (true)
                {
                        c1 = sat1.position () ;
                        c2 = sat2.position () ;
                        currentPosition = theNavigator.compute (c1, c2) ;
                }

        }

}//Transponder
```


Threads Java

- Threads em Java são uma construção simples para implementar objetos concorrentes
- Threads devem incluir um método chamado run(), e ele é inicializado pelo sistema run-time Java
- Objetos ativos tipicamente incluem um loop infinito, para que eles possam sempre estar executando a computação

Um processo de projeto orientado a objetos

- Definir o contexto e os modos de uso do sistema
- Projetar a arquitetura do sistema
- Identificar os principais objetos do sistema
- Desenvolver modelos de projeto
- Especificar interfaces de objetos

Descrição de um sistema meteorológico

Um sistema de mapeamento meteorológico é necessário para gerar mapas meteorológicos regularmente, utilizando dados coletados a partir de estações meteorológicas remotas, sem que seus funcionários estejam presentes, e de outras fontes de dados, como observadores de tempo, balões e satélites meteorológicos. As estações meteorológicas transmitem seus dados ao computador da área em resposta a uma requisição dessa máquina.

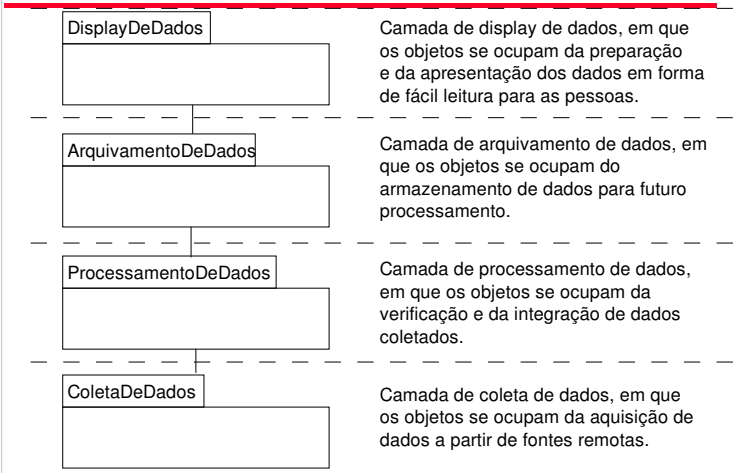
O sistema de computador da área valida os dados coletados e faz a integração dos dados a partir de diferentes fontes. Os dados integrados são arquivados e, com os dados desse arquivo e um banco de dados de mapas digitalizados, é criado um conjunto de mapas meteorológicos locais. Os mapas podem ser impressos para distribuição em uma impressora especial ou ser exibidos em diferentes formatos.

Descrição da estação meteorológica

Uma estação meteorológica é um pacote de instrumentos controlados por software que coleta dados, executa algum processamento de dados e transmite esses dados para processamento adicional. Os instrumentos incluem termômetros de terra e ar, um anemômetro, uma biruta, um barâmetro e um pluviômetro. Os dados são coletados a cada cinco minutos

Quando um comando é enviado para transmitir os dados meteorológicos, a estação processa e resume os dados coletados. Os dados resumidos são transmitidos para o computador de mapeamento quando uma requisição é recebida.

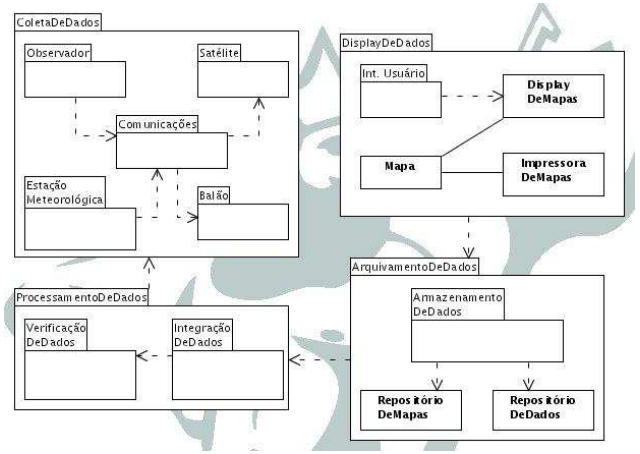
Arquitetura em camadas



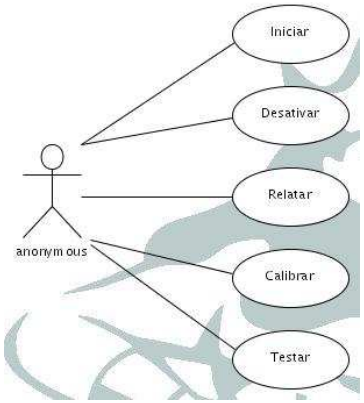
Contexto do sistema e modelos de uso

- Desenvolver uma compreensão dos relacionamentos entre o software que está sendo projetado e o seu ambiente externo
- Contexto do sistema
  - Um modelo estático que descreve outros sistemas no ambiente. Usar um modelo de subsistema para mostrar outros sistemas.
- Modelo de uso do sistema
  - Um modelo dinâmico que descreve como o sistema interage com o seu ambiente. Usar casos de uso para mostrar interações

Subsistemas do sistema de mapeamento meteorológico



Casos de uso para a estação meteorológica



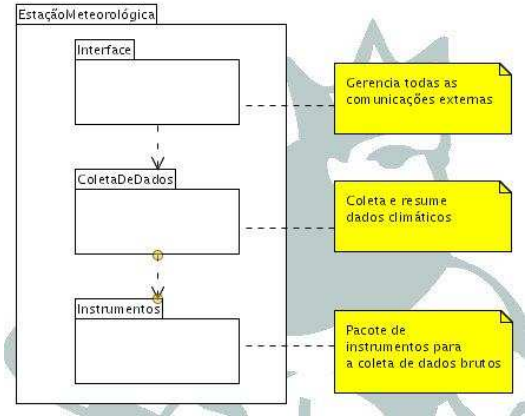
Descrição de caso de uso

Sistema	Estação Meteorológica
Caso de Uso	Relatar
Agentes	Sistema de coleta de dados sobre o clima, Estação meteorológica.
Dados	A estação meteorológica envia para o sistema de coleta de dados climáticos um resumo dos dados sobre o clima, que foram coletados a partir de instrumentos, no período de coleta. Os dados enviados referem-se às temperaturas máximas, mínimas e médias do solo e do ar; à pressão máxima, mínima e média do vento; à precipitação total das chuvas, e à direção do vento, conforme a amostragem a cada intervalo de 5 minutos.
Estímulo	O sistema de coleta de dados sobre o clima estabelece um link de modem com a estação meteorológica e requisita a transmissão dos dados.
Resposta	Os dados resumidos são enviados para o sistema de coleta de dados sobre o clima.
Comentários	Em geral, as estações meteorológicas recebem um pedido de relatório por hora, mas essa freqüência pode diferir de uma estação para outra e ser modificada no futuro.

Projeto de arquitetura

- Uma vez que as interações entre o sistema e o seu ambiente tenham sido compreendidas, essas informações são usadas para projetar a arquitetura do sistema
- A arquitetura em camadas é apropriada para a estação meteorológica
  - *Camada de interface* para manipular comunicações
  - *Camada de coleta de dados* para gerenciar instrumentos
  - *Camada de instrumentos* para coletar dados
- Não deveria haver mais do que 7 entidades em um modelo de arquitetura

Arquitetura da estação meteorológica



Identificação dos objetos

- Identificar objetos (ou classes de objetos) é a parte mais difícil do projeto orientado a objetos
- Não há uma 'fórmula mágica' para identificar objetos. Isso reside na habilidade, experiência e conhecimento de domínio dos engenheiros de sistema
- Identificação de objetos é um processo iterativo. Dificilmente se consegue acertar da primeira vez

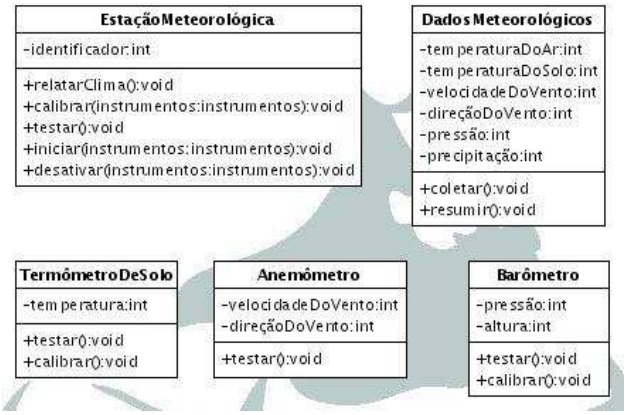
Abordagens para a identificação

- Usar uma abordagem gramatical, baseada em uma descrição do sistema em linguagem natural (usado no método Hood)
- Basear a identificação em coisas tangíveis do domínio da aplicação
- Usar uma abordagem comportamental e identificar objetos com base em quais objetos participam de quais comportamentos
- Usar uma análise baseada em cenários. São identificados os objetos, atributos e métodos em cada cenário

Classes de objetos da estação meteorológica

- Termômetro de solo, Anemômetro, Barômetro
  - Objetos do domínio da aplicação que são objetos de 'hardware' relacionados aos instrumentos do sistema
- Estação meteorológica
  - A interface básica da estação meteorológica para o seu ambiente. Ela, portanto, reflete as interações identificadas no modelo de casos de uso
- Dados meteorológicos
  - Encapsula os dados resumidos dos instrumentos

Classes de objetos do sistema meteorológico



Objetos adicionais e refinamento de objetos

- Usar o conhecimento de domínio para identificar mais objetos e operações
  - Estações meteorológicas devem ter um identificador único
  - Estações meteorológicas são situadas em locais remotos. Então, falhas nos instrumentos têm que ser relatadas automaticamente. Portanto, são necessários atributos e operações para auto-verificação
- Objetos ativos ou passivos
  - Neste caso, os objetos são passivos e coletam dados sob demanda e não autonomamente. Isso introduz flexibilidade, às custas de tempo de processamento do controlador

Modelos de projeto

- Modelos de projeto mostram os objetos e classes de objetos e os relacionamentos entre essas entidades
- Modelos estáticos descrevem a estrutura estática do sistema em termos de classes de objetos e relacionamentos
- Modelos dinâmicos descrevem as interações dinâmicas entre objetos.

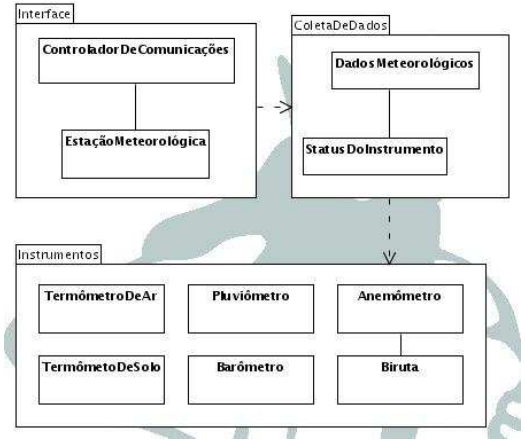
Exemplos de modelos de projeto

- *Modelos de subsistema*, que mostram agrupamentos lógicos de objetos em subsistemas coerentes
- *Modelos de seqüência*, que mostram a seqüência das interações entre objetos
- *Modelos de máquina de estados*, que mostram como objetos individuais trocam seu estado em resposta a eventos
- Outros modelos incluem modelos de caso de uso, modelos de agregação, modelos de generalização, etc.

Modelos de subsistema

- Mostra como o projeto é organizado em grupos de objetos logicamente relacionados
- Em UML, isso é mostrado com o uso de pacotes (*packages*) – uma construção de encapsulamento. Isso é um modelo lógico. A organização real dos objetos no sistema pode ser diferente.

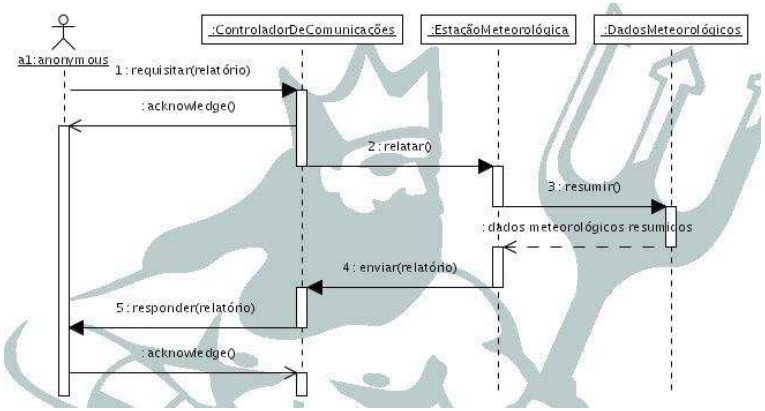
Subsistemas da estação meteorológica



Modelos de Seqüência

- Modelos de Seqüência mostram a seqüência das interações que acontecem entre os objetos
  - Os objetos são organizados horizontalmente no topo do diagrama
  - O tempo é representado verticalmente, de maneira que o diagrama seja lido de cima para baixo
  - As interações são representadas por setas rotuladas. Estilos diferentes de setas representam tipos diferentes de interação
  - Um retângulo fino na linha de vida de um objeto representa o tempo em que o objeto é o objeto controlador no sistema

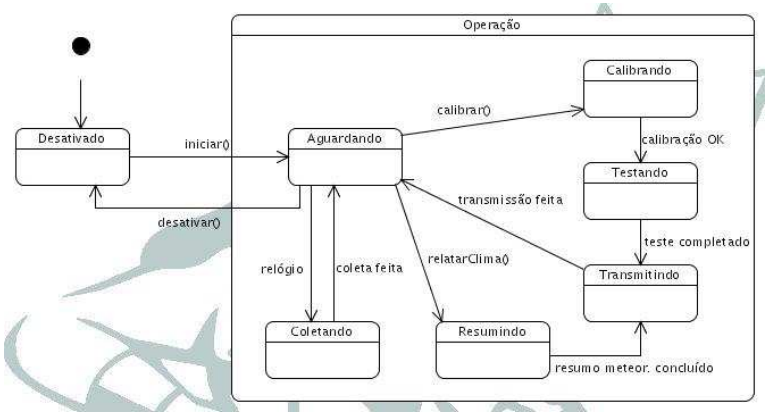
Seqüência da coleta de dados



Diagramas de Estado

- Mostram como os objetos respondem a diferentes solicitações de serviços e as transições de estado disparadas por essas solicitações
  - Se o estado do objeto é Desligado, então ele responde a uma mensagem Iniciar()
  - No estado Aguardando o objeto está aguardando mensagens adicionais
  - Se relatarClima() então o sistema se move para o estado Resumindo
  - Se calibrar(), o sistema se move para o estado Calibrando
  - O sistema entra no estado Coletando quando recebe um sinal do relógio

Diagrama de Estados da estação meteorológica



Especificação de interfaces de objetos

- Interfaces de objetos têm que ser especificadas para que os objetos e outros componentes possam ser projetados em paralelo
- Os projetistas devem evitar projetar a representação da interface, mas devem incluir isso no próprio objeto
- Os objetos podem ter várias interfaces, que são pontos de vista dos métodos fornecidos
- A UML usa diagramas de classe para a especificação da interface, mas isso também pode ser feito em Java

Interface da Estação Meteorológica

```
interface WeatherStation {

        public void WeatherStation () ;

        public void startup () ;
        public void startup (Instrument i) ;

        public void shutdown () ;
        public void shutdown (Instrument i) ;

        public void reportWeather ( ) ;

        public void test () ;
        public void test ( Instrument i ) ;

        public void calibrate ( Instrument i) ;

        public int getID () ;

} //WeatherStation
```

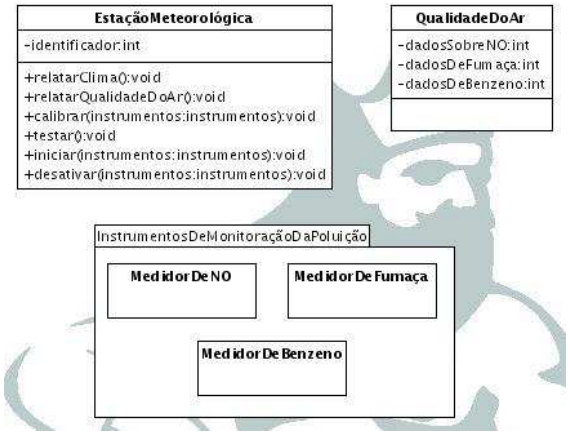

Evolução do Projeto

- Esconder informações dentro do objeto implica que mudanças feitas em um objeto não afetam outros objetos de maneira imprevisível
- Considere-se que dispositivos de monitoramento da poluição devam ser agregados às estações meteorológicas. Eles recolhem amostras do ar e calculam a quantidade de diversos poluentes na atmosfera
- As leituras de poluição são transmitidas com os dados meteorológicos

Mudanças exigidas

- Incluir uma classe de objeto chamada 'Qualidade do ar' como parte de EstaçãoMeteorológica
- Incluir uma operação relatarQualidadeDoAr a EstaçãoMeteorológica. Modificar o software de controle para coletar leituras de poluição
- Incluir objetos representando instrumentos de monitoração da poluição

Monitoração da poluição



Pontos principais

- Projeto OO é uma abordagem na qual cada componentes de projeto têm o seu próprio estado e operações privados
- Os objetos devem ter operações construtoras e de inspeção. Eles fornecem serviços para outros objetos
- Os objetos podem ser implementados seqüencialmente ou concorrentemente
- A UML fornece diversas notações para definir diferentes modelos de objetos

Pontos principais

- Uma série de modelos diferentes pode ser produzida durante um processo de projeto orientado a objetos. Isso inclui modelos do sistema estáticos e dinâmicos
- As interfaces dos objetos deve ser definida com precisão usando, p.ex., uma linguagem de programação como Java
- O projeto OO simplifica a evolução do sistema