Observação: leia atentamente os enunciados e vá fazendo o que está sendo pedido, sem pular etapas. Você deve entender tudo o que está sendo feito. Se necessário, consulte o seu material das aulas de programação. Se você ainda não teve a aula, peça explicações.

1) A classe **Cofrinho** simula um "porquinho" ou um cofrinho digital que só aceita moedas de 10, 25 e 50 centavos. Os objetos desta classe devem armazenar: o nome do dono do cofrinho, a quantidade de moedas de 50 centavos, a quantidade de moedas de 25 centavos e a quantidade de moedas de 10 centavos que foram colocadas no cofrinho. O cofrinho deve ser criado vazio.
A classe deve oferecer as seguintes funcionalidades através dos métodos abaixo:
- ***setNome*** – altera o nome do dono do cofrinho;
- ***getNome*** – retorna o nome do dono do cofrinho;
- ***depositaUmaMoedaDezCentavos*** – o método é acionado a cada moeda de 10 centavos depositada no cofrinho. Atualiza o atributo;

- ***depositaUmaMoedaVinteCincoCentavos*** – o método é acionado a cada moeda de 25 centavos depositada no cofrinho. Atualiza o atributo;
-***depositaUmaMoedaCinquentaCentavos*** – o método é acionado a cada moeda de 50 centavos depositada no cofrinho. Atualiza o atributo;
- ***calculaTotal*** - retorna o valor total em reais, armazenado no cofrinho;
- ***calculaTotal*** - retorna o valor total em dólares, armazenado no cofrinho. O método recebe como parâmetro o valor do dólar. *Obs. Este método é uma sobrecarga do anterior.*

a) Modelagem: desenhe um diagrama de classes usando a notação UML para a classe descrita acima. A partir da descrição você deve identificar o nome da classe, os atributos, os construtores e os métodos.

b) Ainda modelagem: desenhe dois diagramas de objetos usando a notação UML exemplificando dois possíveis objetos desta classe.

c) Após mostrar os diagramas para a professora, programe a classe em Java, seguindo o que está especificado no diagrama de classe e compile-a.

d) Teste-a no blueJ, criando objetos, inspecionando os objetos, aplicando os métodos e inspecionando novamente para verificar se o método provocou alguma mudança no estado dos objetos.

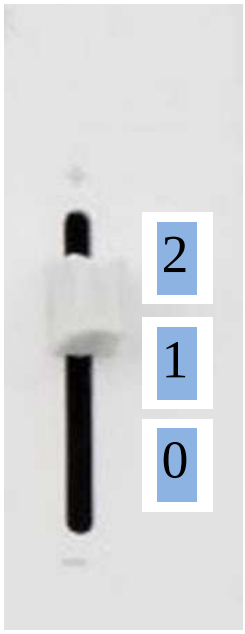
e) Programe a classe **TesteCofrinho**, sem atributos, contendo apenas o método main. Esta classe deve usar a classe **Teclado** (copie-a para dentro do seu projeto, com *add class from file*). A classe de teste deve realizar as operações descritas nos comentários, chamando os métodos adequados:

a. Instanciar um objeto da classe Teclado
b. Criar (instanciar) um cofrinho vazio com o nome do dono lido do teclado.
c. Ex. ***new Cofrinho(t.leString("nome do dono: "))***
d. Criar (instanciar) um segundo cofrinho vazio com o nome do dono lido do teclado.
e. Depositar duas moedas de 10 centavos no primeiro cofrinho.
f. Depositar quatro moedas de 25 centavos no primeiro cofrinho.
g. Depositar três moedas de 50 centavos no primeiro cofrinho.
h. Depositar quatro moedas de 10 centavos no segundo cofrinho.
i. Depositar duas moedas de 25 centavos no segundo cofrinho.
j. Depositar quatro moedas de 50 centavos no segundo cofrinho.
k. Alterar o nome do dono do primeiro cofrinho, lendo o novo nome do teclado
l. Criar um terceiro cofrinho (vazio) para o mesmo dono do segundo
m. Depositar duas moedas de 50 centavos no terceiro cofrinho.
n. Ler do teclado o valor do dólar e armazenar em uma variável.
o. Exibir, na tela o nome do dono do primeiro cofrinho e o valor total contido no cofrinho em reais e em dólares.
p. Exibir, na tela o nome do dono do segundo cofrinho e o valor total contido no cofrinho em reais e em dólares.
q. Exibir, na tela o nome do dono do terceiro cofrinho e o valor total contido no cofrinho em reais e em dólares.
r. Exibir, na tela a diferença, em reais, entre o primeiro e o segundo cofrinho. *Obs. Use o método abs da classe Math para exibir um valor positivo.*

2. Um ventilador analógico pode estar desligado (velocidade 0), em velocidade baixa (1) ou em velocidade alta (2). Para operá-lo, existe um controle deslizante. Assim, estando desligado, ele só pode ir para veloc. 1; desta, pode ir para a 2 ou desligar; estando em 2, só pode diminuir para 1. Isto significa que podemos aumentar a velocidade e diminuir a velocidade do ventilador. Podemos ainda verificar (acessar) a velocidade atual. *Obs. quando não for informada nenhuma velocidade, o ventilador deve ser construído desligado. Do contrário, deve ser construído com a velocidade desejada.*

Diagrama de classe:

VentiladorAnalogico
- velocidade : int
+VentiladorAnalogico( ) **+VentiladorAnalogico(v:int)** **+ getVelocidade( ) : int** **+aumentaVelocidade( ): void** **+diminuiVelocidade( ): void**



a) Desenhe dois diagramas de objetos usando a notação UML exemplificando dois possíveis objetos desta classe:

b) A partir do diagrama de classe, programe a classe em Java e compile-a.

c) Teste-a no blueJ e observe que não podemos impedir que o ventilador fique em um estado inconsistente. Por quê?

3. Programe, em Java, a classe **Calculadora** para oferecer os serviços de uma calculadora muito simples. A calculadora oferece as operações de adição, subtração, multiplicação, divisão inteira, divisão real e resto da divisão inteira para somente dois operandos. Os métodos que fazem adição, subtração, e multiplicação e divisão real devem ser sobrecarregados para aceitar dois operandos int e dois operandos double. Teste-a no blueJ.
   Obs. a classe não tem atributos

4. Faça o diagrama de classe UML e programe a classe **Data** com três atributos inteiros: dia, mês e ano. Faça dois construtores:
   - um construtor que recebe três parâmetros inteiros: o dia, o mês, e o ano;
   - outro construtor com um parâmetro que recebe a data como um número inteiro (na forma AAAAMMDD) e desmembra-a para dar valor aos três atributos da classe. *(Dica: o ano pode ser obtido dividindo-se o número inteiro por 10000 – lembre que a divisão de dois inteiros dá inteiro. Use o code pad do blueJ para verificar e entender este cálculo).*
   A classe deve oferecer os seguintes métodos:
   setDia – recebe um inteiro e configura o atributo dia;
   setMes – recebe um inteiro e configura o atributo mês;
   setAno – recebe um inteiro e configura o atributo ano;

getDia – retorna o atributo dia;
getMes – retorna o atributo mês;
getAno – retorna o atributo ano;
getDataPadrao – devolve a data no formato padrão DD/MM/AAAA (ex: “23/4/2006”)
getDataInvertida – devolve a data na forma de um número inteiro AAAAMMDD (ex: 20060423)
Teste a classe Data no blueJ.