

# ATIVIDADE PRÁTICA PESSOAL 01

```
        }  
        if(parameters.contains("name")){  
            hql += " and p.name = :name ,";  
        }  
        if(parameters.contains("age")){  
            hql += " and p.age = :age ,";  
        }  
        TypedQuery<Person> query = em.createQuery(hql);  
        if(parameters.contains("name")){  
            query.setParameter("name", parameters.get("name"));  
        }  
        if(parameters.contains("age")){  
            query.setParameter("age", Integer.valueOf(parameters.get("age")));  
        }  
        List<Person> list = query.getResultList();  
        return list;  
    }  
}
```



## componente curricular:

# Estrutura de Dados II

# MSc. Fernando Sambinelli

sambinelli@ifsp.edu.br

# Regras

---

- Todo o trabalho feito no sentido do cumprimento das expectativas deste curso deve ser exclusivamente seu
- A colaboração na realização das Atividades Práticas não é permitida, salvo indicação contrária definida na especificação da atividade
- Ver ou copiar o trabalho de outro indivíduo do curso ou retirar material de um livro, site ou outra fonte, mesmo em parte e apresentá-lo como seu próprio constitui desonestidade acadêmica
- Você pode e deve recorrer à Web para obter referências na busca de soluções para as Atividades Práticas, mas não por soluções definitivas para os problemas
- O não atendimento à essas regras serão tratadas com rigor



# Quesitos de Avaliação

---

- **Exatidão:** até que ponto o seu código é consistente com as nossas especificações e livre de bugs?
- **Design:** até que ponto o seu código é bem escrito (escrito claramente, funcionando de forma eficiente, elegante, e / ou lógica)?
- **Estilo:** até que ponto o seu código é legível (comentado e indentado, com nomes de variáveis apropriadas)?



# Calculando o troco correto

- De acordo com o National Institute of Standards and Technology (NIST), um algoritmo guloso é uma técnica que age "sempre realizando a escolha que parece ser a melhor no momento; fazendo uma escolha ótima local, na esperança de que esta escolha leve até a solução ótima global. Algoritmos gulosos encontram a solução global ideal para alguns problemas de otimização, mas eles podem encontrar soluções não-ideais para certos casos de alguns problemas"
- O que tudo isso significa? Bem, suponha que um caixa deve troco a um cliente e no seu cinto se encontram moedas de 25, 10, 5 e 1 centavos. Resolver este "problema" exige uma ou mais retiradas de um ou mais tipos diferentes de moeda

Contar o troco é muito mais fácil com este trocador feito de molas que você usa no seu cinto para despejar moedas em sua mão



# Calculando o troco correto

- Pense em um caixa "guloso", alguém que quer resolver, com cada retirada de uma única moeda, a maior parte possível deste problema. Por exemplo, se algum cliente precisa receber 41¢, o melhor primeiro passo (a melhor escolha imediata, ou local) é a retirada de uma moeda de 25¢ (é "melhor" pois essa retirada nos deixa mais próximo de 0¢ do que qualquer outra retirada de uma única moeda). Note que esse primeiro passo diminui o problema de 41¢ para um problema de 16¢, pois  $41-25 = 16$ . Ou seja, o segundo é um problema semelhante, mas menor.
- Nem precisamos dizer que uma outra retirada de 25¢ seria muito grande (assumindo que o caixa prefere não perder dinheiro), e assim o nosso caixa guloso retiraria uma moeda de 10¢, ficando com um problema de 6¢. Nesse ponto, a ganância pede uma retirada de 5¢ seguida de uma retirada de 1¢, e agora o problema está resolvido. O cliente recebe uma moeda de 25¢, uma moeda de 10¢, uma de 5¢, e uma moeda de 1¢: quatro moedas no total

Contar o troco é muito mais fácil com este trocador feito de molas que você usa no seu cinto para despejar moedas em sua mão



# Calculando o troco correto

- Acontece que essa abordagem gulosa (algoritmo guloso) não só é localmente, mas também globalmente ideal para as moedas do Brasil (e também para as dos EUA ou da União Europeia). Ou seja, enquanto um caixa tem um número suficiente de cada moeda, esta abordagem irá retirar, no final, o menor número de moedas possível
- Quão pequeno é esse número de moedas?
- Escreva um programa que primeiro pede ao usuário quanto de troco ele quer receber e, em seguida, reporta (via printf) número mínimo de moedas com o qual esse troco pode ser dado

Contar o troco é muito mais fácil com este trocador feito de molas que você usa no seu cinto para despejar moedas em sua mão



# Calculando o troco correto

- Você deve ler o valor como um float para que você possa lidar com reais e centavos
- Além disso fique atento pois o padrão de formatação de números decimais em programação utiliza pontos (.) ao invés de vírgulas (,). Em outras palavras, se algum cliente deve receber R\$9.75 (como no caso em que um jornal custa 25¢, mas o cliente paga com uma nota de R\$10), assuma que o input do seu programa será **9.75** e não **9.75 reais** ou **9,75**. No entanto, se algum cliente deve receber exatamente R\$9, assuma que o input do seu programa será 9.00 ou apenas 9, mas, novamente, não R\$9 ou 9,00.
- É claro que, pela natureza dos valores de ponto flutuante (floating point values), o seu programa provavelmente vai trabalhar com inputs como 9.0 e 9.000, assim, você não precisa se preocupar sobre como verificar se o input do usuário é "formatado" como o dinheiro deveria ser
- E você não precisa tentar verificar se o input de um usuário é muito grande para caber em um float. Mas você deve verificar se o input do usuário realmente pode ser visto como um número válido de centavos! Como assim? Garanta que o input do usuário é de fato um valor de ponto flutuante (ou inteiro), mas não que é positivo. Enquanto o usuário não fornecer um valor positivo, o programa deve solicitar novamente ao usuário uma quantidade válida até que o usuário coopere. Aliás, tome cuidado com a imprecisão inerente de valores de ponto flutuante

Contar o troco é muito mais fácil com este trocador feito de molas que você usa no seu cinto para despejar moedas em sua mão



# Calculando o troco correto

- Antes de fazer qualquer conta, você provavelmente vai querer converter a entrada do usuário inteiramente para centavos (a partir de um *float* para um *int*) para evitar pequenos erros que poderiam se somar e tornar-se grandes! Tenha cuidado para não arredondar demais os seus tostões!
- Considere a representação abaixo de como o seu programa deve se comportar; o input de algum usuário está destacado em negrito.

```
Oi. Quanto troco você deve?  
0.41  
4
```

Contar o troco é muito mais fácil com este trocador feito de molas que você usa no seu cinto para despejar moedas em sua mão



# Calculando o troco correto

- Pela natureza dos valores de ponto flutuante, o usuário também poderia ter digitado apenas `.41` representando o mesmo número
- Naturalmente, os usuários mais difíceis, verão algo mais parecido com o abaixo:

```
Oi. Quanto troco você deve?  
-0,41  
Desculpe? Quanto você disse?  
-0,41  
Desculpe? Quanto você disse?  
foo  
Ah... Tente de novo.  
0.41  
4
```

Contar o troco é muito mais fácil com este trocador feito de molas que você usa no seu cinto para despejar moedas em sua mão



# Calculando o troco correto

- Devido a essas exigências (e ao exemplo), o seu código provavelmente vai ter algum tipo de loop. Se, ao testar o seu programa, você se encontrar loopando para sempre, lembre-se que você sempre pode forçar o seu programa a morrer

Contar o troco é muito mais fácil com este trocador feito de molas que você usa no seu cinto para despejar moedas em sua mão



Fonte: Curso CC50 - Harvard University <http://cc50.com.br>

# Resultado Esperado

---

- Compactar o(s) arquivo(s) fonte(s) em formato ZIP e enviar via Edmodo
- Nome do arquivo compactado deve ter o seguinte padrão:  
**APP01.nome\_completo\_aluno.zip**

