Prova 2 - TAP (30/11/2010)

Questão 1 (20 PONTOS)

Explique o conceito de Especialização de Métodos na programação Orientada a Objetos. Cite também uma diferença prática da aplicação desse conceito quando comparado com o que ocorre no paradigma Imperativista.

Questão 2 (30 PONTOS)

Observe o código incompleto em Flavours (Orientado a Objetos) abaixo:

```
flavour menu;
  ivars lista_de_opcoes ...;
defmethod mais_uma_opcao(texto);
  if lista_de_opcoes = undef then
    [ ^texto ] -> lista_de_opcoes;
  elseif lista_de_opcoes = [] then
    [ ^texto ] -> lista_de_opcoes;
  else
    [^^lista_de_opcoes ^texto]->lista_de_opcoes;
  endif;
enddefmethod;
   ...
endflavour;
```

Complete esse código de maneira que ele seja capaz de permitir o seguinte comportamento de execução:

```
: vars m1;
: make_instance([menu]) -> m1;
: m1<-mais_uma_opcao('Consultar saldo');
: m1<-mais_uma_opcao('Efetuar pagamento');
: m1<-mais_uma_opcao('Transferir dinheiro');
: m1<-mais_uma_opcao('Voltar ao menu anterior');
: m1<-captura_opcao;

   Veja as opcoes abaixo:

 1- Consultar saldo.
 2- Efetuar pagamento.
 3- Transferir dinheiro.
 4- Voltar ao menu anterior.

   Escolha uma delas: ? 5

  ATENCAO: opcao invalida! Tente novamente.

   Veja as opcoes abaixo:

 1- Consultar saldo.
 2- Efetuar pagamento.
 3- Transferir dinheiro.
 4- Voltar ao menu anterior.

   Escolha uma delas: ? 3
:
```

Observe que a reação do programa ao usuário diante da escolha de um valor adequado para a faixa de opções (3 no exemplo acima) foi a de aceitar a entrada do usuário e armazená-la de alguma forma (não revelada aqui pois você terá que determiná-la). Note ainda que nenhuma decisão foi tomada com respeito à opção válida (apenas a apresentação das opções e a captura da resposta do usuário foram implementadas).

Para ajudar na entrada de dados, utilize o comando `readline`, que pega os dados de entrada e os copia para uma lista dinâmica. Um exemplo de comportamento do `readline` é o seguinte:

```
: readline() -> lista;
? 3
: lista =>
** [3]
```

Questão 3 (30 PONTOS)

Cite dois princípios ou técnicas do paradigma de Programação Orientada a Objetos que foram utilizados no código da Questão 2. Discuta a razão da adoção de tais princípios ou técnicas e critique (como vantagem ou desvantagem) a maneira com que o código equivalente seria desenvolvido no paradigma de Programação em Lógica (com Prolog).

Questão 4 (20 PONTOS)

O problema clássico da composição de hierarquias de menus com opções tende a ser mais complexo para os programadores do paradigma imperativista. Isso ocorre devido à natureza de seu código de controle, o qual deve ser constituído de um aninhamento de comandos de repetição, tantos quantos forem os itens de dependência da referida hirarquia de opções (altura da árvore de opções). Tais aninhamentos, além de precisarem de estrutura de dados privativa para cada um dos níveis de opção, precisam também do controle cíclico de verificação da reapresentação na memória do menu diante da entrada de uma opção inconsistente.

Explique, em linhas gerais como você faria esse código, reaproveitando ao máximo (por meio de herança) o código da classe `menu` construída no exercício anterior. Não deixe de adicionar fragmentos (incompletos mesmo) de código em Flavours para ilustrar de maneira mais específica a sua solução.

BOA PROVA!!!