Segunda Prova de Técnicas Alternativas de Programação (17/06/08)

Questão 1 (25 PONTOS)

Explique o conceito de Polimorfismo 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 (50 PONTOS)

A estrutura de um "jogador eletrônico" para qualquer tipo de jogo de tabuleiro 'e sempre muito complexa, mas várias simplificações podem ser adotadas. Por exemplo, vamos supor a existência um jogo chamado *FICHAS*. Suas regras são as seguintes:

1. Só há um jogador (não há adversários);
2. O tabuleiro é constituído de cinco casas;
3. Há duas fichas brancas e duas pretas;
4. Em seu estado inicial, as quatro fichas podem ocupar qualquer casa do tabuleiro, ficando sempre uma lacuna vazia. A Figura 1 mostra uma das possíveis configurações de estados iniciais
5. Os dois únicos movimentos permitidos são:
   - fazer com que uma ficha deslize para a lacuna vazia, se a lacuna vazia for adjacente à ficha;
   - fazer com que uma ficha salte para a lacuna vazia, se o salto ocorrer por cima de uma única ficha.
6. O jogo termina quando o jogador conseguir posicionar as duas fichas pretas entre as duas fichas brancas.

Considere o programa INCOMPLETO abaixo, o qual foi escrito em linguagem Flavours (Orientado a Objetos):

```
flavour tabuleiro;
ivras estado;
  ...
endflavour;
flavour ficha;
ivars nome, posicao, ... ;
  defmethod vai_para_tabuleiro_na_casa(T, P);
    lvars T, P;
    P -> posicao;
      ...
  enddefmethod;
  defmethod deslize_para(nova_posicao);
    lvars nova_posicao;
    nova_posicao -> posicao;
      ...
  enddefmethod;
    ...
endflavour;
   ...
```

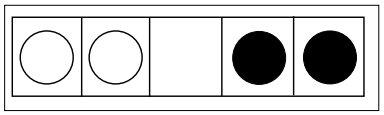

Figura 1: Um estado inicial das fichas.

O código deve ser suficientemente genérico para lidar com a manutenção de vários tabuleiros simulataneamente $(t_1, t_2, ..., t_n)$, cada qual com suas 2 fichas brancas e 2 pretas exclusivas $(fb_{1,1}, fb_{1,2}, fp_{1,1}, fp_{1,2}, ..., fp_{n,2})$. Complete então a definição das classes acima (`tabuleiro`, `ficha` e outras) para que suas instâncias respondam às mensagens que seguem.

```
: vars t1, fb11, fb12, fp11, fp12;
: make_instance([tabuleiro estado [_ _ _ _ _]]) -> t1;
: make_instance([ficha_branca nome b11]) -> fb11;
: make_instance([ficha_branca nome b12]) -> fb12;
: make_instance([ficha_preta nome p11]) -> fp11;
: make_instance([ficha_preta nome p12]) -> fp12;
: fb11 <- quem_e_voce;
Sou uma ficha branca.
: fp11 <- quem_e_voce;
Sou uma ficha preta.
: fb11 <- vai_para_tabuleiro_na_casa(t1, 1);
: fb12 <- vai_para_tabuleiro_na_casa(t1, 2);
: fp11 <- vai_para_tabuleiro_na_casa(t1, 4);
: fp12 <- vai_para_tabuleiro_na_casa(t1, 5);
: t1 <- estado =>
** [fb11 fb12 _ fp11 fp12]
: fb12 <- deslize_para(3);
: t1 <- situacao;
O jogo ainda esta em andamento.
: t1 <- estado =>
** [fb11 _ fb12 fp11 fp12]
: fp11 <- salte_para(2);
...
...
: t1 <- situacao;
O jogo terminou!!!
: t1 <- estado =>
** [fb11 fp11 fp12 fb12 _]
```

Questão 3 (25 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).

BOA PROVA!!!