SCC0204 (Programação Orientada a Objetos) — Prof. Moacir P. Ponti Junior

Trabalho 3

Implemente sua atividade em duplas, sem compartilhar, olhar código de outras duplas, ou buscar na Internet. Procure usar apenas os conceitos já vistos nas aulas. Plágio nesse trabalho gera nota zero para todos os envolvidos.

Programação orientada a objetos concorrente e em rede

Esse trabalho é composto de duas tarefas. A dupla deverá escolher uma das tarefas. Caso realize as duas tarefas, o trabalho irá valer 5 pontos extras.

Será apenas exigido que o trabalho cumpra os objetivos, observando bom uso da programação orientada a objetos (classes, herança, polimorfismo, etc.) e boas práticas de programação. Os detalhes de implementação podem ser definidos por cada dupla.

Tarefa I – threads

Modifique o programa realizado no Trabalho 2 para que permita realizar a execução de ao menos 5 jogos (ao invés de 1) ao mesmo tempo, de forma concorrente. O programa deverá permitir ao usuário carregar a informação de 10 times (2 por jogo). A forma como armazenar e recuperar os dados fica a cargo da dupla. A concorrência será implementada no momento da simulação dos jogos. O usuário deverá escalar cada time (opcionalmente pode haver uma escalação padrão para cada time), e então pressionar um **botão** que dispare a simulação dos 5 jogos.

Tarefa II – rede

Utilizando sockets cliente e servidor, modifique o Trabalho 2 para permitir jogar em rede. O programa deverá ter a opção "jogar em rede" e deverá permitir ao usuário entrar com um IP. Dois usuários deverão jogar, cada um com seu time, e a simulação deverá ocorrer no lado do servidor. Inicialmente um dos usuários executa o programa e "cria" um jogo. A partir disso, o usuário cliente deverá inserir o IP do programa servidor e conectar-se a ele. Após estabelecida a conexão, cada usuário deverá escalar seu time. Ao confirmar a escalação, o programa (cliente ou servidor) envia para o outro lado uma mensagem. Após a escalação de ambos os times, automaticamente o jogo inicia e é realizada a simulação. Nesse momento as escalações de cada time deverão ser visualizadas por ambos os jogadores. A simulação é realizada no servidor que retorna o resultado para o cliente, e exibe também no programa servidor.

Entrega

- O projeto deverá ser entregue apenas pelo Sistema de Submissão de Programas (SSP)[1], escolhendo as seguintes opções:

[1] `http://netuno.icmc.usp.br/ssp01/`

  - No campo “Título do Exercício”: `Trabalho03`;
  - No campo “Linguagem de Programação”: `Zip`.

- Não utilize acentos nos nomes de arquivos.
- Por favor, inclua o número USP da dupla em uma pasta que contém o trabalho e também num arquivo README.
- A detecção de cópia de parte ou de todo código-fonte, de qualquer origem, implicará reprovação direta no trabalho. Partes do código cujas **ideias** foram desenvolvidas em colaboração com outro(s) aluno(s) devem ser devidamente documentadas em comentários no referido trecho. O que **NÃO** autoriza a cópia de trechos de código nem a codificação em conjunto. Portanto, compartilhem ideias, soluções, modos de resolver o problema, mas não o código. Qualquer dúvida entrem em contato com o professor.

Documentação e detalhes de implementação

O código deverá ser obrigatoriamente comentado e documentado, na forma como o exemplo abaixo. Note que é apenas um exemplo, não sendo necessário implementar a função da mesma forma

```
/**
 * Metodo que realiza a simulacao do jogo, por meio de ataques
 * sucessivos e calculo baseado nas habilidades dos jogadores
 *
 * @param listaJogVis - um array com os jogadores visitantes
 * @param listaJogCas - um array com os jogadores da casa
 * @return  tempo em segundos da simulacao
*/
```

Gerar a documentação usando o BlueJ, o NetBeans ou outro software, e incluir a documentação numa pasta separada.

NÃO são usar acentos, cedilha ou caracteres especiais, mesmo nos comentários e docuementação.

Se o programa foi desenvolvido no NetBeans, favor compactar e enviar a pasta completa. Se possível compacte uma pasta com o número USP da dupla.