Universidade Estadual de Maringá – Departamento de Informática
1028 – Linguagens de Programação – Ciência da Computação
1225 – Linguagens de Programação – Informática
Prof. Marco Aurélio

Trabalho do 3º bimestre

1 Introdução

O objetivo deste trabalho é aplicar os recursos de orientação a objetos e concorrência de uma linguagem de programação, na construção de um sistema de rede social.

Além de entregar o código do trabalho, o aluno deverá apresentar o trabalho para o professor. O aluno que demonstrar falta de conhecimento do código, ficará com nota zero.

O trabalho é individual. O compartilhamento de informações entre os alunos é permitido (e aconselhado), mas o compartilhamento de código não é permitido. Trabalhos que tenham código igual serão anulados. Veja a resolução Resolução N° 008/2007-COU para as possíveis sansões disciplinares.

Data de entrega: até o dia 20/10/2011 às 23:00h

Forma de entrega: o trabalho deve ser enviado para o email `malbarbo@gmail.com`, em um arquivo compactado com o nome `codigodisciplina_ra.zip` (substitua `codigodisciplina` pelo código da disciplina e `ra` pelo número do seu RA). O formato de compactação deve ser zip, outros formatos não serão aceitos. Tenha o cuidado de **não** enviar **arquivos compilados**.

Apresentação: cada aluno deve agendar um horário com o professor para fazer a sua apresentação. A data limite para a apresentação é 28/10/2011.

2 Descrição

Criar um sistema de rede social. A especificação das funcionalidades do sistema é feita através de descrições em português e testes funcionais.

Por exemplo, a especificação a seguir descreve (uma parte) da funcionalidade de criação de usuários:

```
Criação de usuários

O sistema deve permitir a criação de novos usuários. Não são permitidos
usuários com o mesmo nome.

> criar-usuario joao
ok
> criar-usuario joao
usuario-ja-existe
```

A primeira linha é o título da especificação. As próximas linhas são o texto da especificação. As linhas iniciadas com `>` marcam o início do teste funcional. As linhas seguintes as marcadas com `>` são as saídas esperadas dos testes funcionais. Observe que os testes funcionais não são detalhados como os testes de unidade. Eles são uma forma executável do texto da especificação, alguns detalhes podem não terem sido especificados.

Os arquivos contendo todas as especificações do sistema estão disponíveis para download na página da disciplina. Além das especificações, está disponível também um testador automático. O testador lê as especificações, executa os testes no sistema e verifica se as saídas estão de acordo com as especificadas. O testador também realiza um teste de concorrência.

O sistema deverá ser escrito como um servidor que ouve por conexões na porta 1234. A comunicação entre o servidor e os clientes deve ser feita através de mensagens de texto codificadas em `utf-8`. Cada cliente que faz uma conexão com o servidor envia um comando, recebe a resposta e encerra a conexão. O servidor recebe o comando, processa o comando, envia a resposta para o cliente e encerra a conexão. Os comando enviados pelos clientes podem ter apenas uma linha. Não existe limite de linhas para a resposta do servidor. O servidor deve poder atender mais do que um cliente ao mesmo tempo.

Além dos comandos que aparecem nas especificações, o servidor deve responder a um comando especial chamado `resetar`. Este comando é utilizado pelo testador para informar ao servidor que ele deve apagar todos os dados armazenados, para que outra seção de teste possa começar. O servidor não precisa armazenar os dados de forma permanente, apenas durante a execução.

Você pode escolher uma das seguintes linguagens para fazer o trabalho: Ada, C# ou Java. A principio C++ também poderia ser usado, mas como a linguagem não oferece mecanismos de controle de concorrência, ela foi desconsiderada.

O programa deve poder ser compilado e executado em um sistema Linux. Um arquivo executável chamado `iniciar` deve ser fornecido com o código do programa. Quando este arquivo for executado, ele deve compilar o programa e iniciar o servidor. Veja o exemplo em Java que acompanha o testador.

3 Avaliação

O trabalho será avaliado de acordo com os critérios:

- Corretude e completude: o sistema tem que passar em todos os testes funcionais
- Boas práticas de programação: o código deve estar bem escrito e organizado; os recursos da linguagem devem ser usados corretamente
- Modelagem dos objetos do sistema
- Controle da concorrência

4 Desafios

Cada desafio a seguir vale 1,0 extra na nota do bimestre. Você pode fazer mais do que um desafio e ganhar mais pontos. Para poder ganhar os pontos extras, o aluno tem obter no mínimo a nota 8,0 no trabalho.

Desafio 1: Escrever o código em Ada.

Desafio 2: Escrever a especificação (de maneira semelhante as especificações já escritas) e a implementação para as funcionalidades de menções e busca simples (conforme o funcionamento do twitter).

Desafio 3: Escrever uma interface gráfica semelhante a do twitter.

5 Dicas

As informações abaixo são apenas dicas. Analise-as individualmente e verifique quais são interessantes.

- Execute o testador e veja o próximo comando que deve ser implementado. Escreva o código necessário para passar no teste (não pense em outras funcionalidades neste momento, apenas no comando atual). Só avance para o próximo comando, quando o comando atual estiver funcionando.
- Antes de implementar um comando, leia e entenda as especificação do comando.
- Em um primeiro momento, escreva o programa para passar nos testes funcionais, sem se preocupar com o controle de concorrência. Quando o seu programa estiver passando nos testes funcionais, **salve uma cópia** (ou melhor, use um sistema de controle de versão) do seu programa e adicione o controle de concorrência. O mesmo vale para implementar os desafios.
- Para implementar o controle de concorrência de competição, veja quais os dados são compartilhados pelas tarefas concorrentes, e garanta o acesso exclusivo a estes dados.
- Consulte os amigos e o professor se você estiver com dificuldades.