



Programação com Objectos/Projecto de Programação com Objectos/Repositório CVS

From Wiki**3

< Programação com Objectos | Projecto de Programação com Objectos

AVISOS - Avaliação em Época Normal

[Collapse]

Esclarecimento de dúvidas:

- Consultar sempre o corpo docente atempadamente: presencialmente ou através do endereço oficial da disciplina [1].
- Não utilizar fontes de informação não oficialmente associadas ao corpo docente (podem colocar em causa a aprovação à disciplina).
- Não são aceites justificações para violações destes conselhos: quaisquer consequências nefastas são da responsabilidade do aluno.

Requisitos para desenvolvimento, material de apoio e actualizações do enunciado (ver informação completa em Projecto de Programação com Objectos):

- O material de apoio é de uso obrigatório e não pode ser alterado.
- Verificar atempadamente (mínimo de 48 horas antes do final de cada prazo) os requisitos exigidos pelo processo de desenvolvimento.

Processo de avaliação (ver informação completa em Avaliação do Projecto):

- Datas: **2022/10/04 12:00** (inicial); **2022/10/21 12:00** (intercalar); **2022/11/04 12:00** (final); **2022/11/04 (early bird) 2022/11/07 (normal)** (teste prático).
- **Todas as entregas são cruciais para o bom desenvolvimento do projecto, sendo obrigatórias: a não realização de uma entrega implica a exclusão da avaliação do projecto e, por consequência, da avaliação da disciplina.**
- Verificar atempadamente (até 48 horas antes do final de cada prazo) os requisitos exigidos pelo processo de avaliação, incluindo a capacidade de acesso ao repositório CVS.
- **Apenas se consideram para avaliação os projectos existentes no repositório CVS oficial.**
- Trabalhos não presentes no repositório no final do prazo têm classificação 0 (zero) (não são aceites outras formas de entrega). Não são admitidas justificações para atrasos em sincronizações do repositório. A indisponibilidade temporária do repositório ou de outros materiais, desde que inferior a 24 horas, não justifica atrasos na submissão de um trabalho.
- A avaliação do projecto pressupõe o compromisso de honra de que o trabalho correspondente foi realizado pelos alunos correspondentes ao grupo de avaliação.
- **Fraudes na execução do projecto terão como resultado a exclusão dos alunos implicados do processo de avaliação.**

Material de Uso Obrigatório

[Collapse]

As bibliotecas **po-uilib** e o conteúdo inicial do CVS são de **uso obrigatório**:

- po-uilib (classes de base) po-uilib-202209081626.tar.bz2 (não pode ser alterada) - javadoc
- prr-core (classes do "core") (via CVS) (deve ser completada -- os nomes das classes fornecidas não podem ser alterados)
- prr-app (classes de interacção) (via CVS) (deve ser completada -- os nomes das classes fornecidas não podem ser alterados)

A máquina virtual, fornecida para desenvolvimento do projecto, já contém todo o material de apoio.

Uso Obrigatório: Repositório CVS

Apenas se consideram para avaliação os projectos existentes no repositório CVS oficial.

Trabalhos não presentes no repositório no final do prazo têm classificação 0 (zero) (não são aceites outras formas de entrega). Não são admitidas justificações para atrasos em sincronizações do repositório. A indisponibilidade temporária do repositório, desde que inferior a 24 horas, não justifica atrasos na submissão de um trabalho.

Contents

- 1 Localização
- 2 Modo de Acesso
- 3 Conteúdo e Estrutura do Repositório
- 4 Cuidados a ter na Utilização do Repositório CVS
 - 4.1 NÃO FAZER
 - 4.2 FAZER
 - 4.3 Observações importantes
- 5 "Receita" para a Entrega de Diagramas UML

Os repositórios CVS para os projectos (um por grupo) já contêm versões iniciais do projecto.

⚠ O USO DO REPOSITÓRIO CVS É OBRIGATÓRIO.

⚠ NÃO SERÃO CONSIDERADOS PROJECTOS QUE NÃO ESTEJAM NO CVS.

💡 Pequeno guia para leitores perplexos: CVS Crash Course.

Localização

Os repositórios estão disponíveis no AFS em:

- `/afs/ist.utl.pt/groups/leic-po/po22/cvs`

O conteúdo do repositório pode ser manipulado com o comando `cvs` ou com qualquer outro cliente compatível (e.g. *eclipse*).

⚠ Esta directoria não deve ser utilizada directamente.

Modo de Acesso

No texto que se segue:

- `###` designa o número de grupo com três dígitos, e.g., 012
- `ISTID` designa o identificador de aluno, no formato "ist123456"

Assim, os valores possíveis para a variável de ambiente `CVSROOT` (ou pela opção `-d` do comando "cvs") são:

- `/afs/ist.utl.pt/groups/leic-po/po22/cvs/###` (disponível em clientes AFS e utilizador com token válido para a célula ist.utl.pt)
- `:ext:ISTID@sigma.ist.utl.pt:/afs/ist.utl.pt/groups/leic-po/po22/cvs/###` (disponível via SSH; verificar que `CVS_RSH=ssh`)

⚠ Alunos que ainda não visitaram o self-service da DSI, devem fazê-lo, para activação dos serviços relevantes (AFS e Shell, pelo menos). Sem isso, não conseguirão desenvolver ou entregar o projecto, reprovando à disciplina.

Conteúdo e Estrutura do Repositório

O código fornecido é de uso obrigatório e deve ser completado.

Cada projecto é constituído por dois módulos: **project/prr-core** e **project/prr-app**.

Conteúdo inicial do repositório CVS para o módulo prr-core

[Collapse]

```
prr-core
├── Makefile
├── README.md
├── src
│   └── prr
│       ├── exceptions
│       │   ├── ImportFileException.java
│       │   ├── MissingFileAssociationException.java
│       │   ├── UnavailableFileException.java
│       │   └── UnrecognizedEntryException.java
│       ├── Network.java
│       ├── NetworkManager.java
│       └── terminals
│           └── Terminal.java
```

Estes módulos podem ser obtidos em conjunto fazendo checkout de **project** (ver abaixo).

```

prr-app
├── Makefile
├── README.md
├── src
│   └── prr
│       └── app
│           ├── App.java
│           ├── clients
│           │   ├── DoDisableClientNotifications.java
│           │   ├── DoEnableClientNotifications.java
│           │   ├── DoRegisterClient.java
│           │   ├── DoShowAllClients.java
│           │   ├── DoShowClient.java
│           │   ├── DoShowClientPaymentsAndDebts.java
│           │   ├── Label.java
│           │   ├── Menu.java
│           │   ├── Message.java
│           │   └── Prompt.java
│           ├── exceptions
│           │   ├── DuplicateClientKeyException.java
│           │   ├── DuplicateTerminalKeyException.java
│           │   ├── FileOpenFailedException.java
│           │   ├── InvalidTerminalKeyException.java
│           │   ├── Message.java
│           │   ├── UnknownClientKeyException.java
│           │   └── UnknownTerminalKeyException.java
│           ├── lookups
│           │   ├── DoShowAllCommunications.java
│           │   ├── DoShowClientsWithDebts.java
│           │   ├── DoShowClientsWithoutDebts.java
│           │   ├── DoShowCommunicationsFromClient.java
│           │   ├── DoShowCommunicationsToClient.java
│           │   ├── DoShowTerminalsWithPositiveBalance.java
│           │   ├── DoShowUnusedTerminals.java
│           │   ├── Label.java
│           │   ├── Menu.java
│           │   ├── Message.java
│           │   └── Prompt.java
│           ├── main
│           │   ├── DoOpenFile.java
│           │   ├── DoSaveFile.java
│           │   ├── DoShowGlobalBalance.java
│           │   ├── Label.java
│           │   ├── Menu.java
│           │   ├── Message.java
│           │   └── Prompt.java
│           ├── terminal
│           │   ├── DoAddFriend.java
│           │   ├── DoEndInteractiveCommunication.java
│           │   ├── DoPerformPayment.java
│           │   ├── DoRemoveFriend.java
│           │   ├── DoSendTextCommunication.java
│           │   ├── DoShowOngoingCommunication.java
│           │   ├── DoShowTerminalBalance.java
│           │   ├── DoSilenceTerminal.java
│           │   ├── DoStartInteractiveCommunication.java
│           │   ├── DoTurnOffTerminal.java
│           │   ├── DoTurnOnTerminal.java
│           │   ├── Label.java
│           │   ├── Menu.java
│           │   ├── Message.java
│           │   ├── Prompt.java
│           │   └── TerminalCommand.java
│           └── terminals
│               ├── DoOpenMenuTerminalConsole.java
│               ├── DoRegisterTerminal.java
│               ├── DoShowAllTerminals.java
│               ├── Label.java
│               ├── Menu.java
│               ├── Message.java
│               └── Prompt.java

```

O repositório já tem a seguinte estrutura:

- **prr-core** (directoria principal da biblioteca com a funcionalidade sem interface com o utilizador)
- **prr-core/Makefile** (makefile secundária, semelhante à disponibilizada na aplicação bancária)
- **prr-core/src** (directoria onde reside o código do "core", à semelhança do que acontece na aplicação bancária)
- **prr-app** (directoria correspondente à aplicação; contém a interface com o utilizador)
- **prr-app/Makefile** (makefile secundária, semelhante à disponibilizada na aplicação bancária)
- **prr-app/src** (directoria onde já reside o código da interface textual, tal como na biblioteca equivalente da aplicação bancária)

Esta estrutura já contém algumas classes parcialmente implementadas (**pr-r-core**) (que devem ser adaptadas) e outras parcial ou completamente implementadas (**pr-r-app**) (algumas das quais não podem ser alteradas).



A estrutura do repositório não pode ser alterada.



O código fornecido em **pr-r-core** e **pr-r-app** tem de ser completado.



Os nomes das classes fornecidas em **pr-r-core** e **pr-r-app** não podem ser alterados.



As seguintes classes e interfaces não podem ser alteradas: **App.java**, todos as **Label.java**, todos as **Menu.java**, todos as **Message.java**, todos os **Prompt.java** e todas as excepções fornecidas.

Cuidados a ter na Utilização do Repositório CVS

NÃO FAZER

- **cvs init** (não é necessário: o repositório já existe)
- **cvs import** (não é necessário: o projecto já existe)
- **cd /afs/ist.utl.pt/groups/leic-po/po22/cvs** (ou qualquer outra manipulação directa dos ficheiros do repositório: toda a interacção com o repositório deve ser limitada ao comando "cvs")

FAZER

Antes de tudo o mais, fazer cópias de segurança (vulgo "backups") de tudo o que já foi feito.

O repositório já contém uma versão preliminar do projecto e devem ser dados os seguintes passos, para continuar a desenvolver o projecto:

1. **cvs co project** (permite criar localmente uma directoria controlada pelo CVS, com o nome **project**, com a estrutura apresentada acima, contendo os ficheiros do projecto);
2. Editar/adicionar material do/ao projecto;
3. Caso sejam criados ficheiros ou directórios novos, fazer **cvs add ficheiro1 ficheiro2 ... ficheiro3** (adicionar cada ficheiro novo ao projecto no CVS);
4. **cvs update** (antes de enviar alteração locais para o repositório, verificar se há actualizações a incorporar na cópia local e que não foi esquecido nenhum ficheiro importante: voltar a 3 até terem sido todos adicionados)
5. **cvs commit** (enviar alterações para o repositório)

Observações importantes

- No ponto 3 acima, é importante notar que quando se quer adicionar o conteúdo de uma (sub)directoria, deve ser primeiro adicionada a própria directoria e, só depois, o seu conteúdo.
- Não adicionar a biblioteca **po-uilib** ao projecto (serão removidas e quaisquer alterações perdidas -- isto pode causar problemas de compilação e perda de nota).
- Pessoas sem área no AFS, ou sem login activado, não poderão aceder ao conteúdo do repositório: para activar estes acessos, consultar o self-service do CIIST (DSI). Em caso de dificuldade, consultar o corpo docente.
- A estrutura do projecto no CVS deve ser mantida como apresentado e cada biblioteca do projecto deve ser, em geral, semelhante, em estrutura, às apresentadas para a aplicação bancária.
- Em caso de dúvidas, consultar o corpo docente.

Agradece-se a comunicação de eventuais problemas.

"Receita" para a Entrega de Diagramas UML

Tal como indicado no método de avaliação, os 3 diagramas a entregar -- **UML-po-uilib.pdf**, **UML-pr-r-app.pdf**, **UML-pr-r-core.pdf** -- devem ser colocados na directoria **uml** do projecto no repositório CVS.

Os comandos abaixo explicam como realizar essa operação a partir de um projecto no repositório (desde obter uma cópia local até completar a entrega). O passo de obter a cópia local pode ser omitido se o projecto já tiver sido obtido anteriormente. O passo de adicionar novos ficheiros pode ser omitido se as versões dos diagramas corresponderem a actualizações de adições anteriores. Nos exemplos dos comandos abaixo, **ist123456** é o ISTid da pessoa que realiza as operações e **199** é o grupo dessa pessoa (estes dois valores variam de pessoa para pessoa e de grupo para grupo).

- Obter o projecto do repositório (o projecto fica na directoria local **project**). Assume-se que os comandos subsequentes são dados dentro da directoria **project** (notar **cd project** abaixo):

```
cvsv -d :ext:ist123456@sigma.tecnico.ulisboa.pt:/afs/ist.utl.pt/groups/leic-po/po22/cvs/199 co project
cd project
```

- Colocar os ficheiros com os diagramas na directoria **project/uml** (ficheiros novos ou actualizações). Esta operação depende de onde estão os diagramas ou de como foram digitalizados a partir da versão manuscrita (recorda-se que apenas são aceites diagramas manuscritos, devidamente acompanhados com a declaração de autoria). O comando apresentado é apenas um exemplo (assume que os diagramas estão originalmente na directoria principal do utilizador).

```
cp ~/UML-po-uilib.pdf ~/UML-prr-app.pdf ~/UML-prr-core.pdf uml
```

- Adicionar os ficheiros novos ao CVS (no comando a seguir, assume-se que os 3 ficheiros são adicionados, mas pode adicionar-se um de cada vez):

```
cvsv add uml/UML-po-uilib.pdf uml/UML-prr-app.pdf uml/UML-prr-core.pdf
```

- Pode realizar-se neste ponto um pedido de actualização da cópia local (caso a outra pessoa do grupo também esteja a fazer alterações -- se não é o caso, este passo é opcional):

```
cvsv update
```

- Finalmente, enviam-se os ficheiros novos para o repositório CVS (a mensagem descreve o commit e evita a abertura de um editor):

```
cvsv commit -m "Adicionados diagramas UML."
```

Categories: **Projecto de PO PO Ensino**