FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Laboratório de Programação Orientada por Objetos, MIEIC, 2015/16
2º Trabalho Prático (Projeto Integrado)

PYS5 Teeter

1 Instruções

1.1 Objectivos

O objectivo do trabalho é desenvolver um programa em Java para dispositivos móveis Android (*smartphone* ou *tablet*), devidamente documentado e testado, aplicando técnicas de desenho e programação orientada por objetos.

Todos os programas incluem as seguintes componentes de base:

- interface gráfica para o utilizador;
- manipulação de ficheiros.

Todos os programas incluem pelo menos uma das seguintes componentes mais avançadas:

- animação de objetos gráficos;
- funcionamento em rede.

1.2 Atribuição

O trabalho é realizado em grupos de, preferencialmente, 2 alunos da mesma turma. Não podem existir dois grupos da mesma turma a realizar o mesmo tema. Os temas serão atribuídos nas aulas práticas de 5 a 11 de Abril de 2016.

1.3 Entrega

1.3.1 *Checkpoint* intermédio (peso 10%)

Nas aulas práticas da semana de 9 a 13 de Maio de 2016 será efetuado um *checkpoint* intermédio focado nos seguintes elementos:

- desenho do programa:
    - desenho da estrutura de *packages* e classes do programa, esquematizada através de diagramas UML, com a responsabilidade de cada classe documentada;
    - desenho de aspetos relevantes de comportamento do programa, preferencialmente esquematizados através de diagramas dinâmicos em UML (estados, sequência ou atividades),
    - identificação de padrões de desenho utilizados;

    - identificação de *frameworks*, bibliotecas e tecnologias existentes que serão utilizadas (exemplo: libGDX);
- desenho da GUI e funcionalidades
    - identificação de principais funcionalidades
    - desenho da GUI, através de esboços da interface (protótipo *throw away*) ou imagens da interface já criada em Java (protótipo evolutivo);
- desenho de testes
    - lista de testes ou exemplos a usar para efeito de testes.

Notas: No seguimento de dúvidas levantadas, prestam-se alguns esclarecimentos adicionais sobre o *checkpoint* intermédio:

- A não ser que o docente das aulas práticas o solicite para mais facil apreciação fora da aula, este *checkpoint* não corresponde a nenhuma submissão de artefactos, mas apenas a uma verificação na própria aula dos artefactos produzidos relativamente aos pontos indicados acima.
- Diagramas UML: podem ser desenhados em papel ou computador; o diagrama de classes só é necessário para a camada lógica, e só com atributos mais importantes (podendo opcionalmente indicar alguns métodos importantes).
- Padrões: indicar os que foram aplicados e aonde (pode ser com notas associadas aos diagramas de packages e classes).
- Desenho da GUI: esboços dos ecrãs mais importantes em papel ou imagens retiradas da web de ecrãs a reproduzir também servem;
- Testes: basta lista dos objetivos dos testes a realizar pela interface com o utilizador (em português ou inglês).

1.3.2 Entrega final (peso 35%)

A entrega final do projeto deve ser feita até às 23h55m do dia 6 de Junho de 2016 (para permitir depois a demonstração e discussão do trabalho durante essa semana), incluindo:

- Código fonte do programa em Java (incluindo testes unitários);
- Código executável do programa;
- Páginas HTML com documentação Javadoc;
- Ficheiro com modelo UML criado no Enterprise Architect;
- Relatório de projeto em PDF contendo manual de utilização e documentação de conceção do projeto (incluindo imagens de diagramas UML, referência a padrões de desenho utilizados, etc.).

1.4 Código de conduta

O trabalho sujeito a avaliação tem de ser original. O projeto pode incluir algum código não original, que deve estar devidamente assinalado (com referência aos autores originais / url). Espera-se que os membros da equipa contribuam equitativamente para o projeto; se não for

esse o caso, indicar no relatório final a percentagem de contribuição de cada elemento para o projeto (totalizando 100%) e a justificação para contribuições não equitativas.

1.5 Avaliação

A avaliação do trabalho prático será essencialmente focada na correta implementação das funcionalidades pedidas com base nas técnicas de programação orientada a objetos. Entre estas, destaca-se o seguinte:

- Uso adequado de técnicas de abstração de dados, encapsulamento, modularidade, polimorfismo, e herança;
- Correta identificação e aplicação de padrões de desenho OO (e.g., observer, singleton, interpreter, composite, etc.);
- Cobertura e significância adequada no desenvolvimento de testes unitários e de integração;
- Utilização adequada de UML para apoio à conceção e documentação do programa.

Itens e pesos:

- Demonstração do funcionamento do programa (funcionalidades, usabilidade, robustez, funcionamento em rede ou animação) - 30%;
- Estruturação do código (estrutura de *packages*, estrutura de classes, utilização de técnicas de orientação por objetos, padrões de desenho, comentários javadoc) - 25%;
- Testes (testes automáticos, testes manuais) - 10%;
- Acompanhamento semanal - 10%;
- Relatório (manual de utilização, documentação em UML, etc.) - 15%;
- Checkpoint intermédio - 10%.

2 Temas

Segue-se uma lista de temas propostos. Também se aceitam sugestões de temas dos alunos, que terão de ser aprovadas pelos docentes.

SPE/Mostra UP

Temas com dificuldade média a elevada, em que os utilizadores (visitantes) usam os seus dispositivos móveis para participar por turnos num jogo ou atividade cujo estado vai sendo mostrado num monitor.

M1 Jogo de cartas

Cada utilizador vê as suas cartas e efetua as suas jogadas a partir do seu dispositivo móvel em Android. A ‘mesa’ é mostrada num monitor desktop. As aplicações ‘cliente’ que correm nos dispositivos móveis comunicam com uma aplicação ‘servidora’ (também em Java) que gere a mesa e o display.

M2 Jogo de dominó

Cada utilizador vê as suas peças e efetua as suas jogadas a partir do seu dispositivo móvel em Android. A ‘mesa’ é mostrada num monitor desktop. As aplicações ‘cliente’ que correm nos dispositivos móveis comunicam com uma aplicação ‘servidora’ (também em Java) que gere a mesa e o display.

M3 Batalha naval

Cada utilizador vê a sua frota e efetua as suas jogadas a partir do seu dispositivo móvel em Android. O estado atual do jogo é mostrada num monitor desktop. As aplicações ‘cliente’ que correm nos dispositivos móveis comunicam com uma aplicação ‘servidora’ (também em Java) que gere a mesa e o display.

M4 Bowling

Cada utilizador efetua as suas jogadas (orientação e força do lançamento) a partir do seu dispositivo móvel em Android. A situação do jogo é mostrada num monitor desktop. As aplicações ‘cliente’ que correm nos dispositivos móveis comunicam com uma aplicação ‘servidora’ (também em Java) que gere a plataforma e o display. Usar libGDX para garantir portabilidade e para gerir a física do jogo.

M5 Jogo de bilhar (proposto por alunos)

Cada utilizador efetua as suas jogadas (orientação e força do taco) a partir do seu dispositivo móvel em Android. A ‘mesa’ é mostrada num monitor desktop. As aplicações ‘cliente’ que correm nos dispositivos móveis comunicam com uma aplicação ‘servidora’ (também em Java) que gere a mesa e o display. Usar libGDX para garantir portabilidade e para gerir a física do jogo.

Shoot-em-up!

SU1 Jogo tipo Raiden

Clássico jogo de arcade (naves espaciais). http://en.wikipedia.org/wiki/Raiden_(video_game)

SU2 Jogo dos asteróides

Desenvolver um programa com interface gráfica que permita ao utilizador jogar aos asteroides. Neste jogo, o utilizador encontra-se numa nave espacial e pode apontar e disparar uma arma para destruir asteroides em movimento, evitando ser atingido por algum asteroide (caso em que perde). A nave desloca-se no espaço contíguo usando retro-propulsores que podem ser acionados e um leme que permite rodar a nave em ambas as direções. Assume-se que existe um leve atrito no espaço devido ao lixo espacial. Suportar níveis de dificuldade variáveis e estatísticas dos melhores jogadores.

BoardGame/Strategy

STR1 Jogo do Monopólio

Possivelmente em rede.
http://en.wikipedia.org/wiki/Monopoly_(game)

STR2 Jogo do Risco

Possivelmente em rede.
http://en.wikipedia.org/wiki/Risk_(game)

STR3 Batalha naval

Escrever um programa com interface gráfica que permita ao utilizador jogar à batalha naval contra o computador ou contra outro jogador em rede.

Platform/Puzzle

PP1 Boulder Dash

http://en.wikipedia.org/wiki/Boulder_Dash

PP2 Rescue Rover

http://en.wikipedia.org/wiki/Rescue_Rover

PP3 Bubble shooter

http://www.silvergames.com/bubble-shooter

PP4 Jogo do Tetris

Implementar o jogo do Tetris. O jogo deve permitir 3 modos de jogo: 1 jogador, 2 jogadores, 1 jogador vs. computador. Suportar níveis de dificuldade (velocidades) variáveis e estatísticas dos melhores jogadores.

PP5 Jogo do Pacman

Desenvolver programa que permite jogar o pacman. Deve ser possível definir o número de fantasmas, gerar aleatoriamente o labirinto e o número de pastilhas. Suportar níveis de dificuldade variáveis e estatísticas dos melhores jogadores.

Physics Games

PYS1 Jogos de bilhar

Desenvolver um programa com interface gráfica que permita simular o jogo de bilhar. Os parâmetros do jogo devem ser configuráveis (atrito, massa, etc.). Deve ser possível jogar dois jogadores à vez.

PYS2 Jogos de snooker

Desenvolver um programa com interface gráfica que permita simular o jogo de snooker. Os parâmetros do jogo devem ser configuráveis (atrito, massa, etc.). Deve ser possível jogar dois jogadores à vez.

PYS3 Bowling

Desenvolver um programa com interface gráfica que permita simular o jogo de bowling. Os

parâmetros do jogo devem ser configuráveis (atrito, massa, etc.). Deve ser possível jogar dois jogadores à vez (no mesmo dispositivo ou em dispositivos diferentes). Possibilidade de guardar ranking.

PYS4 Tiro aos pratos

Desenvolver um programa com interface gráfica que permita simular um jogo de tiro aos pratos. Os parâmetros do jogo devem ser configuráveis (massa, etc.). Deve ser possível jogar dois jogadores à vez.

PYS5 Teeter

https://play.google.com/store/apps/details?id=com.gfagame.teeter

Nota: Para iniciar o desenvolvimento de aplicações para Android na versão de Eclipse instalada na FEUP é necessário em Window->Preferences->Android->SDK location indicar C:\Programs\Android\android-sdk. Em alternativa a Eclipse, pode ser usado o Android Studio (http://developer.android.com/tools/studio/index.html).

AMA / HSF / JGB / JPF/ NHF