**Projeto final da disciplina (nota AV3, Implementação)**

Prezados alunos, chegou a hora do trabalho final da disciplina de Programação Orientada a Objetos! Vamos às regras:

* **Cálculo da nota**:
  + **Definição de papéis (1,0 pts)**
    - O grupo deverá **obrigatoriamente** formalizar um papel para cada um dos integrantes (**prazo disso é até o dia 03/11, conforme cronograma**). Uma pessoa (ex: Astrogildo) pode, por exemplo, ficar responsável pelo diagrama de classes, enquanto outras duas pessoas (ex: Creuzilene e Asclepíades) ficam encarregadas do código, e outra pessoa fica encarregada dos testes. No entanto, mesmo com os papéis rigorosamente identificados, nada impede que, por exemplo, alguém da parte de implementação ajude também no diagrama, e assim por diante… Caso o grupo esqueça de formalizar uma tarefa para algum integrante, o grupo ficará sem a pontuação referente à definição de papéis;
  + **Uso dos conceitos estudados na disciplina de Programação Orientada a Objetos (3,0 pts)**:
    - Além da implementação, o grupo deverá buscar fazer uma boa modelagem do sistema, devendo então utilizar os seguintes conceitos:
      * Ao menos uma classe abstrata e a ocorrência de Herança em relação a essa classe abstrata criada pelo grupo (0,5 pts);
      * Ao menos uma Interface especificada pelo grupo e uma classe que implemente esta interface; (0,5 pts);
      * Ao menos uma sobrescrita de método (dica: implemente um toString() na classe principal do seu programa) (0,5 pts);
      * Para todas as classes que tenham atributos: todos devem ser privados, e devem ter métodos get e set públicos implementados (0,5 pts);
      * Ao menos uma ocorrência de Tratamento de Exceção (0,5 pts);
      * Interface Gráfica com o Usuário (0,5 pts);
  + **Diagrama de Classes fidedigno com o código compilável (2,0 pts)**
    - Todas as classes e interfaces implementadas deverão estar presentes no diagrama de classes, e vice e versa. Além disso, o diagrama deve estar conceitualmente consistente e fidedigno com o código, que deve ser compilável;
  + **Testes executados pelo professor (2 pts)**:
    - Esta tarefa servirá como termômetro para medir a qualidade do sistema implementado. O professor criará 5 casos de testes para cada grupo, e, durante o período reservado para as apresentações do trabalho, pedirá ao grupo para executá-los… Cada caso de teste valerá 0,4 pontos;
  + **Progresso entre sprints (2 pts)**: em cada semana, o professor organizará sprints com os grupos, para apurar o andamento do trabalho, e se houve progresso de uma semana para a outra. **Cada sprint deverá fornecer evidências de progresso do trabalho entre uma semana e outra, possivelmente solucionando pendências de backlog**. Cada sprint que tiver registrado progresso valerá 0,5 ponto (já que são 4 sprints, o total pode alcançar 2,0);
* **Acompanhamento do trabalho**:
  + O professor organizará sprints semanais para monitorar e acompanhar o desenvolvimento do projeto para cada grupo. Como dito no tópico anterior, as sprints deverão fornecer avanços em relação à sprint anterior. A cada interação com o professor, novos requisitos ou problemas podem surgir, e eles deverão ser incluídos no trabalho (no documento e na implementação);
* **Disponibilização do trabalho**:
  + O código-fonte e o relatório do trabalho deverão ser disponibilizados no *github* ([GitHub: Let’s build from here · GitHub](https://github.com/)), em uma conta pertencente a algum integrante do grupo, ou em uma conta criada pelo próprio grupo;
* **Sobre a avaliação do trabalho**:
  + **Os trabalhos serão avaliados estritamente com base nos critérios estabelecidos aqui**. Fiquem tranquilos que não haverá comparação entre trabalhos. Por exemplo: suponha que o grupo do Pacman fez um trabalho vistosamente maravilhoso, com uma mega-interface gráfica tridimensional, trilha sonora, e em que o personagem exibe expressões faciais e etc, mas não fez o diagrama de classes… Este grupo ficará sem 2 pontos automaticamente, independente do quão impactante tenha sido a implementação. Já um outro grupo, suponha, que fez o Sokobox com interface simples e satisfez tudo o que foi exigido nas regras, com o sistema passando por todos os casos de testes… Este grupo poderá ficar com 10,0;
  + O grupo **não poderá copiar trabalhos** de colegas de outros semestres, podendo sofrer penalizações na nota final… no máximo, poderá utilizar outros códigos como inspiração;
  + O grupo poderá utilizar o auxílio de chatbots (como o ChatGPT), mas deverá documentar no relatório como utilizou o chatbot (quais foram as perguntas feitas a ele), e devem tecer críticas acerca das soluções dadas pelo chatbot;
  + Cada **grupo terá uma nota**, e esta **será a nota de TODOS** **os integrantes do grupo** na AV3;

Agora, vamos às sugestões de temas para o trabalho de vocês…

Compilador de HTML, XML ou JSON (proibidas APIs de terceiros)

Este projeto consiste em implementar um compilador para alguma das linguagens acima. Serão no máximo 3 grupos, e não poderão ocorrer repetições. Regras adicionais:

* Serão permitidas apenas as seguintes APIs em cada classe: *java.util*, *java.lang*, *javax.swing*, *java.awt*, e qualquer outra implementada pelo próprio grupo (enviar o código desta API no próprio projeto). Isso significa que o grupo não pode dar *import* de nenhuma outra API fora as listadas aqui. Caso o grupo escolha implementar este projeto, o **uso de APIs externas poderá incorrer em penalização** de 50% na nota total;

Editor de textos (proibidas APIs de terceiros)

Este projeto consiste em implementar um editor de textos com alguma sofisticação, em que o usuário pode escolher a opção de fornecer syntax highlighting para palavras específicas (no caso de editores de linguagens de programação, por exemplo), ou mesmo alterar o tipo da letra (Consolas, Times New Roman, Arial e etc). Este editor deve ser capaz de salvar um arquivo de texto com o conteúdo e a formatação.

* Serão permitidas apenas as seguintes APIs em cada classe: *java.util*, *java.lang*, *javax.swing*, *java.awt*, e qualquer outra implementada pelo próprio grupo (enviar o código desta API no próprio projeto). Isso significa que o grupo não pode dar *import* de nenhuma outra API fora as listadas aqui. Caso o grupo escolha implementar este projeto, **uso de APIs externas poderá incorrer em penalização** de 50% na nota total;

Editor de imagens

Este projeto consiste em implementar um editor de imagens com alguma sofisticação, permitindo que sejam regulados brilho, contraste, matiz, níveis de vermelho, verde e azul, e etc…

* Serão permitidas apenas as seguintes APIs em cada classe: *java.util*, *java.lang*, *javax.swing*, *java.awt*, e qualquer outra implementada pelo próprio grupo (enviar o código desta API no próprio projeto). Isso significa que o grupo não pode dar *import* de nenhuma outra API fora as listadas aqui. Caso o grupo escolha implementar este projeto, o **uso de APIs externas poderá incorrer em penalização** de 50% na nota total;

Loja Virtual, Sistema Bancário ou qualquer outro sistema que envolva armazenamento de dados

Neste tipo de projeto, você não necessariamente precisa realizar a integração com algum banco de dados (pode utilizar os conceitos que vimos em sala de aula que dizem respeito à manipulação de arquivos).

Jogo 2D (Pacman, Sokobox, Cobrinha, Labirinto ou etc)

Este projeto consiste em implementar um jogo 2D com interface simples. Este jogo deve ser extensível para qualquer número de fases, e o cenário de cada fase pode ser especificado em um arquivo de texto que será lido pelo seu programa.

**Dicas úteis para o desenvolvimento deste projeto**

Você com certeza achará muito útil a interface KeyListener… :-)

Algumas observações (podem existir outras também… e estas podem fazer parte dos requisitos):

* É interessante que os cenários (fases) sejam armazenados em arquivos de texto (strings) que serão lidos na inicialização do seu programa/jogo. Desta forma, você pode, de forma simples, estender o seu jogo para várias fases diferentes;
* Você deve ficar atento aos obstáculos das fases para que o personagem não os ultrapasse, mesmo após pressionar uma tecla direcional;
* Os movimentos dos fantasmas (no caso do Pacman) podem ser implementados seguindo uma lógica randômica…
* No caso do Sokobox, se o personagem estiver adjacente a alguma caixa e ele se mover do lado oposto, a caixa também deverá se mover, caso não haja uma parede depois;