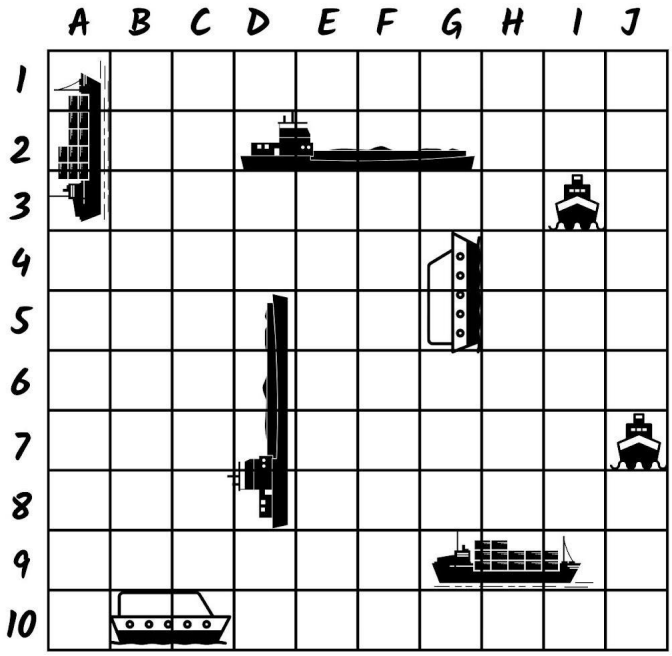
Trabalho 1

PROBLEMA:
Implementar um jogo de batalha naval orientado a objetos em Python.

ESCOPO DO DESENVOLVIMENTO:
Implementar uma versão simplificada de um jogo de batalha naval que permita ao jogador jogar contra o computador.

O jogador é cadastrado com nome e data de nascimento. A pontuação total do jogador e o histórico das jogadas também devem ser armazenados.

O jogo consiste em posicionar embarcações de diferentes tamanhos em um oceano. Um oceano é um espaço bidimensional referenciado por Números e Letras (N,L). Por exemplo, uma embarcação pode ser colocada na posição (2,D) a (2,G) do oceano, conforme a figura abaixo.



O jogador possui o seu próprio oceano e distribui suas embarcações em posições (N,L) nesse oceano. Um jogador não pode visualizar o oceano do oponente (computador). Em cada turno, o jogador atira em uma determinada posição do oceano (N,L) do oponente e recebe a informação se o tiro acertou alguma embarcação.

Existem diferentes tipos de embarcações: bote (ocupa uma posição), submarino (ocupa duas posições), fragata (ocupa 3 posições) e porta-aviões (ocupa 4 posições).

Dinâmica do jogo:

1. O jogador recebe 8 embarcações (3 botes, 2 submarinos, 2 fragatas e 1 porta-aviões) e posiciona essas embarcações em posições do oceano
2. Em cada rodada, o jogador indica em qual posição deseja atirar e após o tiro, recebe a informação se o tiro acertou alguma embarcação do oponente. Se o jogador acertar o tiro, ele pode atirar novamente. Cada tiro acertado vale um ponto. Quando todas as partes de uma embarcação foram acertadas a embarcação naufraga. Uma embarcação completa afundada vale mais 3 pontos
3. O jogador perde se todas as suas embarcações forem afundadas e ganha se conseguir afundar todas as embarcações do oponente

O jogador deve jogar contra o computador. Ou seja, no início do jogo, as posições das embarcações no oceano do computador devem ser geradas de forma aleatória. Da mesma forma, em cada rodada, as posições dos tiros do computador são também geradas de forma aleatória.

Deve ser possível listar todos os jogos dos jogadores cadastrados e mostrar a classificação dos jogadores de acordo com a pontuação.

Considere as seguintes regras:

1. O jogo é monousuário, ou seja, o jogador sempre joga contra o computador.

2. O tamanho do oceano (uma matriz) é definido pelo jogador no início do jogo, tendo um limite máximo de acordo com o tamanho da tela.

3. O programa pode fornecer feedback visual minimalista para acertos, erros e embarcações naufragadas na grade do oceano usando, inicialmente, representações baseadas em texto.

4. O programa deve apresentar o total de pontos atual do usuário e do oponente (computador).

5. A classificação geral dos jogadores cadastrados se dá pelo total de pontos acumulados.

ESCOPO:
Para simplificar este trabalho, o jogo implementado visa fornecer uma experiência de jogo básica, sem o uso de inteligência artificial.

Para este tema padrão, serão considerados:

- **cadastros**: jogador, embarcações e oceano.
- **registros**: jogo (data/hora, jogador, oceanos do jogador e do computador, pontuações e jogadas).
- **relatórios**: histórico de jogadas e pontuações do jogador, classificação dos jogadores.

ENTREGAS:

Parte 1: Deve ser postado um arquivo ZIP por equipe no Moodle até o dia **29/09/2023** às 18:00hs, contendo:

- Um documento em formato PDF com a listagem das telas (menus) do sistema e como serão utilizadas e a divisão das atividades do trabalho entre os membros da equipe;
- Todas as figuras com os diagramas de classes, seguindo a notação UML 2. No caso de mais de 10 classes, deve ser elaborado um diagrama de classes por funcionalidade do sistema, englobando: controladores, classes de apresentação/telas e as entidades.

Parte 2: Deve ser postado um arquivo ZIP por equipe no Moodle até o dia **16/10/2023** às 18hs, contendo:

- Código fonte completo do sistema orientado a objetos em Python.
- Figuras contendo os diagramas de classes atualizados seguindo a notação UML 2. No caso de mais de 10 classes, deve ser elaborado um diagrama de classes por funcionalidade do sistema, englobando: controladores, classes de apresentação/telas e as entidades.

APRESENTAÇÕES / DEFESAS DOS TRABALHOS:
A apresentação do trabalho será realizada **presencialmente** por cada grupo em sala de aula ou laboratório. Na apresentação deve ser **demonstrado o sistema desenvolvido em execução**. Cada um dos membros do grupo deverá **apresentar a parte do código-fonte que desenvolveu**, explicando o código-fonte implementado. A nota de cada membro da equipe será **individual**, dependendo da sua **participação individual no desenvolvimento do trabalho e na apresentação**.
A **participação individual** do aluno no trabalho pode ser **comprovada** ao professor por meio da listagem dos *commits* na ferramenta de controle de versões do código-fonte (GitHub ou similar), ou outra forma de relatório que comprove de forma inequívoca que os códigos foram elaborados pelo aluno.
O aluno que não estiver presente na apresentação (salvo motivo regimentalmente justificável) receberá nota zero no trabalho.

CRITÉRIOS DE AVALIAÇÃO:

- Cadastro dos jogadores, embarcações e oceanos, contemplando para cada um: inclusão, exclusão, alteração e listagem (2,0 pontos).
- Registro das jogadas e jogos: inclusão, exclusão e listagem (1,5 pontos).
- Emissão dos relatórios (1,5 pontos).

Além desses critérios funcionais, também serão avaliados:

- Documentação das telas (menus) do sistema e como serão utilizadas e a divisão das atividades (0,5 ponto)
- Qualidade, uso da notação e consistência do diagrama de classes (0,5 ponto)
- Utilização correta de: associação, agregação e composição (1,0 ponto)
- Utilização correta do MVC (1,0 ponto)
- Utilização correta de: herança e classes abstratas (1,0 ponto)
- Tratamento de todas as exceções (1,0 ponto)