Lista de Exercícios 1 - (5,0 pontos)

Regras

- A atividade ficará aberta no Blackboard até a data limite de entrega;
- A atividade deverá ser entregue **exclusivamente** pelo Blackboard;
- Os alunos que não tiverem acesso ao Blackboard deverão entrar em contato com o professor;
- A submissão da atividade deve ser feita em um **único arquivo zip**. Este arquivo deve ter o nome completo do aluno, sem caracteres especiais, separado por *underlines*, ex: `Joao_da_Silva.zip`. Este zip deve conter apenas 2 arquivos **.java**, **exatamente** com os nomes `Reacao.java` e `Personagem.java`, cada um com o código-fonte da classe correspondente;
- Caso seu código não compile ou apresente erros de execução (*crash*, loop infinito, etc.), a nota daquele exercício automaticamente é **zerada**;
- A atividade deve ser feita de maneira **individual**. Caso dois alunos submetam códigos iguais ou **muito semelhantes**, será considerado plágio. Consequentemente, **ambos** os alunos receberão **zero** na atividade inteira;
- A correção do código será feita de forma automática, avaliando o comportamento do mesmo com testes de várias entradas diferentes. Você terá acesso a alguns testes para lhe ajudar durante o desenvolvimento, mas os testes da correção serão diferentes;
- Siga cuidadosamente as instruções de cada exercício, atentando-se para corresponder **com exatidão** aos requisitos de cada exercício e as validações necessárias;
- Teste seu código também com entradas diferentes, para ter certeza de que está atendendo a todos os critérios acima;

Exercício 1 (2,5 pontos)

Desenvolva uma classe chamada `Reacao` que representa um recipiente onde ocorre uma reação química no formato:

$$\mathrm{xA} + \mathrm{yB} \rightarrow \mathrm{C}$$

Em outras palavras, $\mathbf{x}$ unidades do elemento A e $\mathbf{y}$ unidades do elemento B resultam em uma única unidade do elemento C. A reação só acontece quando o recipiente é agitado.

A classe deverá possuir os seguintes métodos **públicos**:

- Um construtor que recebe $\mathbf{x}$ e $\mathbf{y}$, ou seja, as quantidades dos elementos A e B, respectivamente, necessárias para gerar um elemento C
- `adicionarA(int)`, adiciona uma quantidade de elemento A ao recipiente
- `adicionarB(int)`, adiciona uma quantidade de elemento B ao recipiente
- `agitar()`, realiza a reação, produzindo o máximo de C possível. Atente-se que é possível que reste unidades de A ou B
- `getC()`, retorna a quantidade de elemento C no recipiente

Use atributos e métodos privados à vontade para auxiliar a implementação.

Exercício 2 (2,5 pontos)

Desenvolva uma classe denominada `Personagem` que irá representar um personagem dentro de um MOBA/RPG.

Um personagem tem um **nível**, que no início de uma partida é sempre 1. Esse nível pode ser melhorado ao receber pontos de experiência. A cada 100 pontos de experiência que um personagem recebe, o seu nível é incrementado em 1. O nível máximo que um personagem pode alcançar é 25.

Todo personagem tem quatro habilidades (*skills*/magias), e cada habilidade tem seu próprio nível de melhoria. Todas as 4 habilidades iniciam a partida com melhoria em 0 (não podendo ser usada). A cada nível do personagem (*incluindo o nível 1*), ele pode escolher melhorar uma das habilidades, incrementando o nível dela em 1. As três primeiras habilidades têm um nível máximo de 4. A quarta habilidade é especial (*ultimate*), tendo um nível máximo de 3, e apenas podendo ser melhorada quando o personagem atingir o nível 6.

Um personagem tem uma quantidade pré-determinada de pontos de mana, que são consumidos conforme usa suas habilidades. Cada habilidade tem um custo de mana, que é um custo base (diferente para cada habilidade) multiplicado pelo nível atual daquela habilidade.

A classe deverá possuir os seguintes métodos **públicos**:

- Um construtor que recebe 5 argumentos, representando a quantidade de mana máxima do personagem, seguido pela quantidade de mana base de cada uma das quatro habilidades. A mana do personagem inicialmente é igual a mana máxima.
- `adicionarXP(int)`: adiciona uma quantidade de pontos de experiência ao personagem
- `getNivel()`: retorna o nível atual do personagem
- `melhorarHabilidade(int)`: melhora uma das quatro habilidades do personagem (indexada por 0). Retorna um booleano indicando se foi possível melhorá-la
- `usarHabilidade(int)`: ativa a habilidade do personagem (indexada por 0), consumindo mana no processo. Retorna um booleano indicando se foi possível usar a habilidade
- `consumirPocao()`: recarrega a mana do personagem em 350. A mana total não pode ultrapassar a mana máxima.

Use atributos e métodos privados à vontade para auxiliar a implementação.