Prova II

Disciplina: Laboratório de Programação Orientada a Objeto - POO

Período: 3°

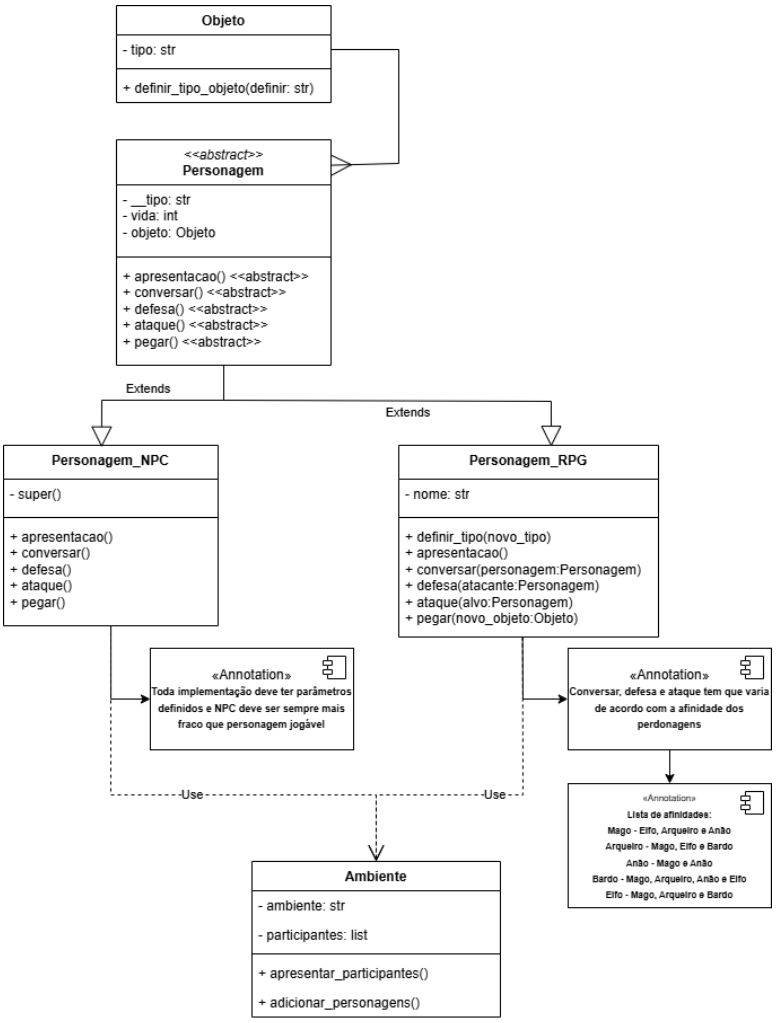
Valor da Prova: 07 Pontos

Professor: Diego Ramos Inácio

Curso de Engenharia de Software

Município: Saquarema – RJ

Prova de Programação Orientada a Objetos – RPG com Base em Diagrama UML



Questão 1 – Classes e Atributos Com base no diagrama fornecido, implemente as classes: Objeto, Personagem (abstrata), Personagem_NPC, Personagem_RPG e Ambiente. Respeite todos os atributos apresentados, bem como suas visibilidades. A classe Personagem deve ser abstrata e conter os atributos _tipo, vida e objeto, além de servir como base para as outras classes. A classe Ambiente deve conter o nome do ambiente e uma lista de participantes. (ponto 1,75)

Questão 2 – Métodos e Regras de Implementação: Implemente todos os métodos descritos no diagrama. Na classe Personagem_NPC, os métodos devem ser simples e sempre mais fracos que os da classe Personagem_RPG. Já os métodos da classe Personagem_RPG devem permitir ataques, defesas, conversas, definição de tipo e troca de objetos com lógica condicional de acordo com os tipos dos personagens. (ponto 1,75)

Questão 3 – Abstração, Polimorfismo e Visitor – Classe Visitante: Implemente os conceitos de abstração, polimorfismo e Visitor (classe visitante) no seu código. A abstração deve estar presente por meio da classe Personagem, que é abstrata e contém métodos obrigatórios a serem implementados pelas classes filhas. O polimorfismo deve ser aplicado na redefinição dos métodos

ataque, defesa, conversar e pegar, com comportamentos diferentes em Personagem_RPG e Personagem_NPC. O padrão de projeto conhecido como Visitor, também chamado de classe visitante, deve ser utilizado na lógica de ataque e defesa. Esse padrão permite que a lógica de interação entre personagens seja centralizada fora das classes principais. Assim, crie uma classe visitante que represente o ataque, e use esse objeto para "visitar" os personagens-alvo. Ao visitar, a classe visitante deve identificar o tipo do personagem e aplicar a regra de afinidade: se o atacante for do tipo que representa uma fraqueza para o defensor, o dano causado será maior. Essa verificação deverá ocorrer dentro da classe visitante, simulando a lógica de interações entre diferentes tipos de personagens sem sobrecarregar as classes dos próprios personagens. (ponto 1,75)

Questão 4 – Execução e Testes: Crie um pequeno script que instancie dois personagens jogáveis com tipos diferentes, um NPC, e os adicione a um ambiente. Em seguida, realize ataques, defesas, conversas e trocas de objeto entre eles, apresentando os participantes e testando todas as interações principais do sistema. (ponto 1,75)