POO C#

START

PLANO AULA

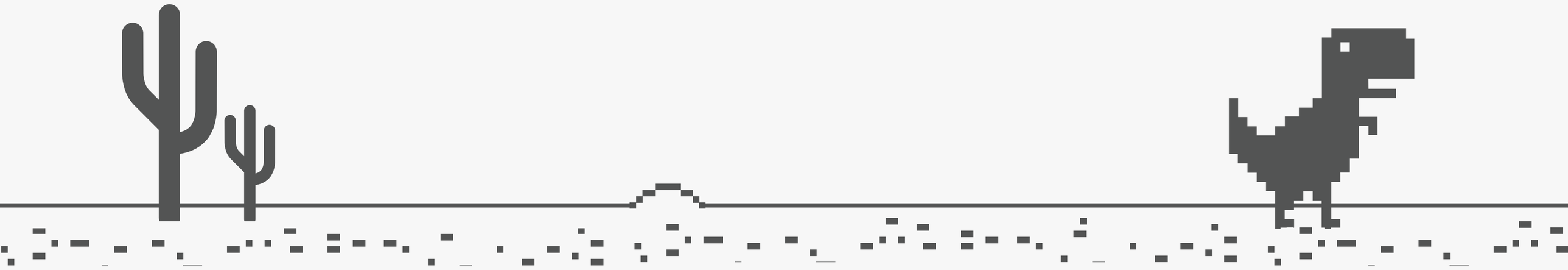
THERE ARE 3 DIFFERENT
TYPES OF GAMES!

REVISÃO CONCEITOS

COMPOSIÇÃO EM C#

EXERCÍCIOS

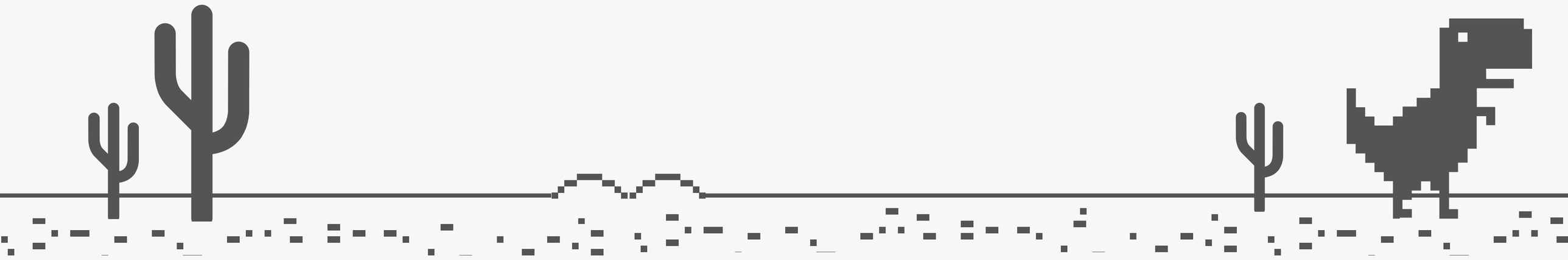
REVISANDO POO



O QUE É POO?

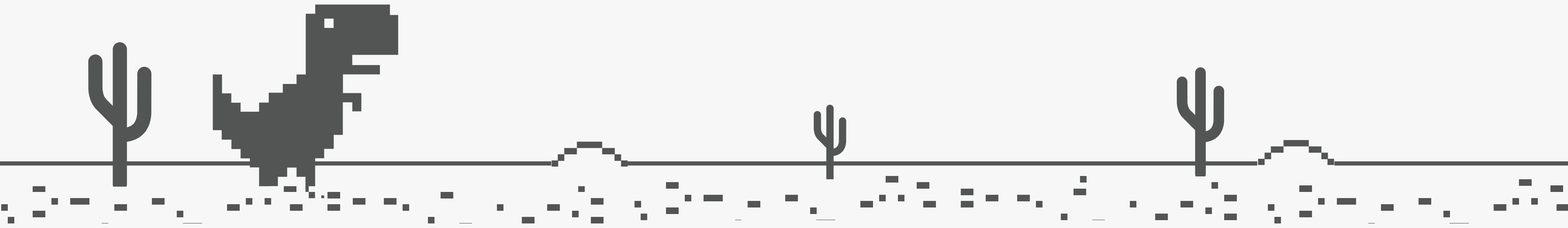
Pilares:
Abstração, Encapsulamento, Herança, Polimorfismo.

Extra: Composição e Coleções



C# VS. C++: O QUE MUDA?

Enquanto o C exige controle manual de memória, o C# automatiza essa tarefa com o Garbage Collector, além de apresentar uma sintaxe mais limpa e simplificada, especialmente no uso de propriedades para encapsulamento.

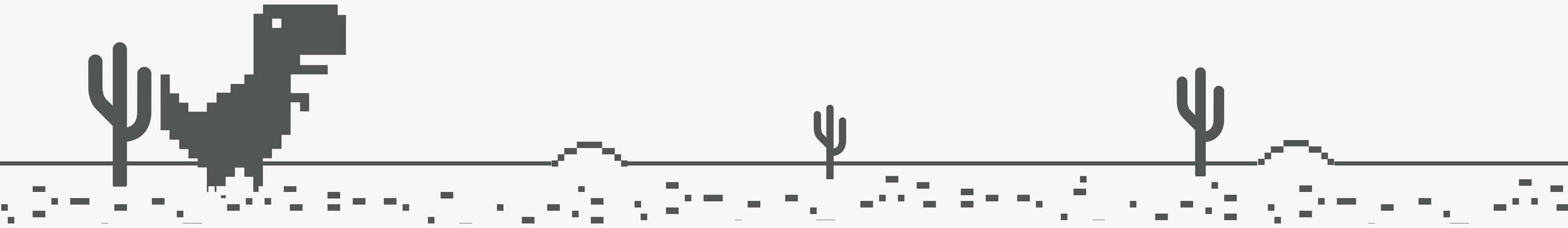


HERANÇA, POLIMORFISMO E ENCAPSULAMENTONO C#

Herança: Classe filha adquire métodos e atributos da classe pai. Usada para reuso de código.

Polimorfismo: O mesmo método tem comportamentos diferentes e classes distintas. Permite tratar objetos de classes filhas como se fossem o tipo pai.

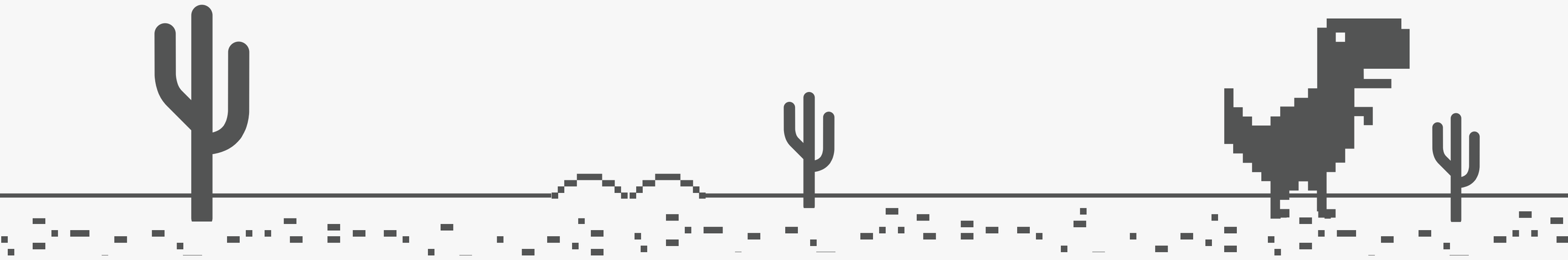
Encapsulamento: Protege os dados internos do objeto. Controla o acesso via public, private e protected.



AGREGAÇÃO

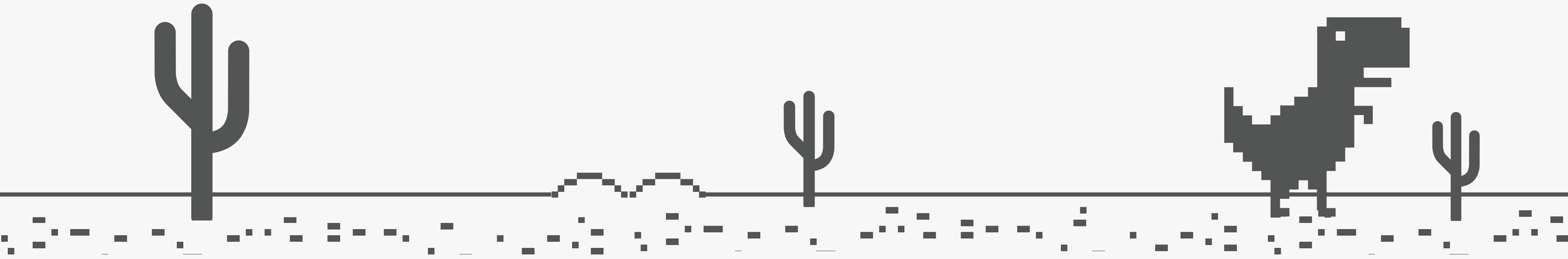
Agregação (Relacionamento "Tem"):

- **É um tipo de Composição mais fraco.**
- **Representa uma relação "todo-parte" onde as partes podem existir independentemente do todo.**
- **Exemplo: Um Carro agrega um Motor. O motor pode ser removido e usado em outro carro.**



COMPOSIÇÃO

- **Composição (Forte): A parte não pode existir sem o todo.**
- **Exemplo: Um Carro compõe um Chassi. Se o carro for destruído, o Chassi não sobrevive sozinho.**



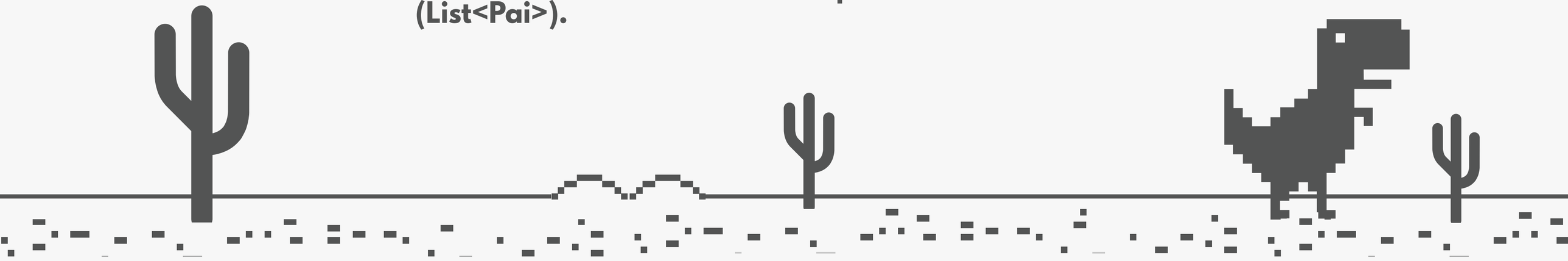
CONSTRUTORES E COLEÇÕES EM C#

Construtores:

- **Método automático que inicializa o objeto.**
- **Garante que o objeto nasça em um estado válido.**
- **Chamado usando new.**

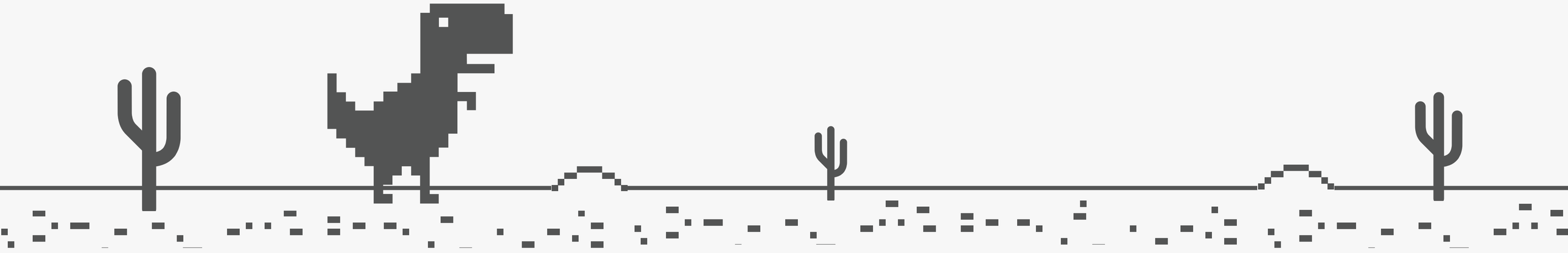
Coleções (Listas e Vetores):

- **Usadas para armazenar vários objetos.**
- **List<T>: Coleção dinâmica (tamanho variável).**
- **Polimorfismo: Permite armazenar objetos de diferentes classes filhas em uma lista do tipo da classe mãe (List<Pai>).**



[ANSWER]

EXERCÍCIOS



EXERCÍCIO 1: A SOCIEDADE DO ANEL

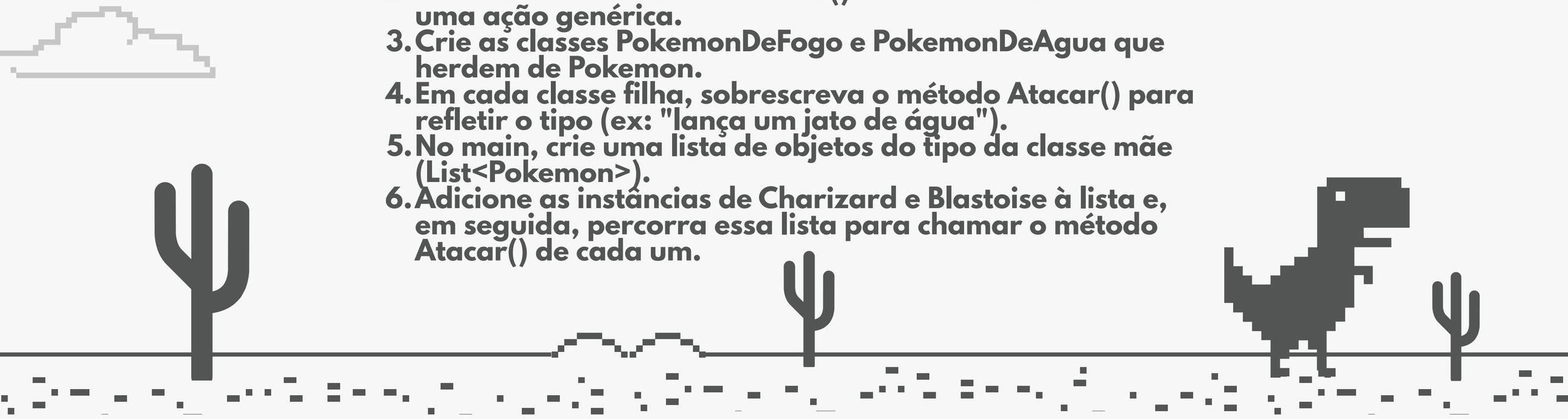
Cenário: Na Sociedade do Anel, cada membro possui um nome, uma raça e uma função. Tarefa:

1. **Crie a classe MembroDaSociedade.**
2. **Defina atributos para o nome, a raça e a função do membro. (todos privados)**
3. **Crie um Construtor que obrigue a inicialização desses três atributos ao criar um novo membro.**
4. **Crie um método Descrever() que imprima as informações completas do membro.**
5. **No main, crie as instâncias de Aragorn (Humano, Guardião) e Legolas (Elfo, Arqueiro) usando o construtor e chame o método Descrever() em ambos.**

EXERCÍCIO 2: EVOLUÇÕES E TIPOS

Cenário: Em Pokémon, o Charizard é um tipo de Pokémon de Fogo, e o Blastoise é um tipo de Agua. Todos são capazes de usar um ataque elemental. Tarefa:

1. **Crie uma classe base chamada Pokemon. Adicione um atributo Nome.**
2. **Crie um método virtual Atacar() na classe Pokemon com uma ação genérica.**
3. **Crie as classes PokemonDeFogo e PokemonDeAgua que herdem de Pokemon.**
4. **Em cada classe filha, sobrescreva o método Atacar() para refletir o tipo (ex: "lança um jato de água").**
5. **No main, crie uma lista de objetos do tipo da classe mãe (List<Pokemon>).**
6. **Adicione as instâncias de Charizard e Blastoise à lista e, em seguida, percorra essa lista para chamar o método Atacar() de cada um.**

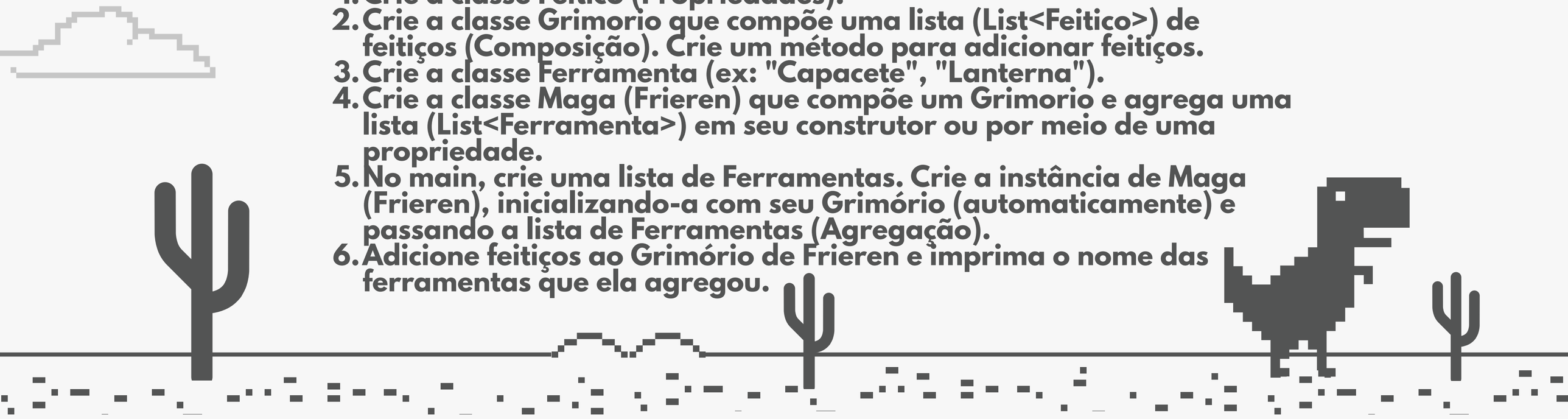


EXERCÍCIO 3: ARMAZENAMENTO MÁGICO

Cenário: A maga Frieren armazena seus feitiços em um livro mágico, mas também possui um conjunto de Ferramentas de Sobrevivência que ela apenas utiliza (agregação). O Grimório é parte essencial dela (composição), enquanto o conjunto de ferramentas pode ser descartado ou compartilhado (agregação).

Tarefa:

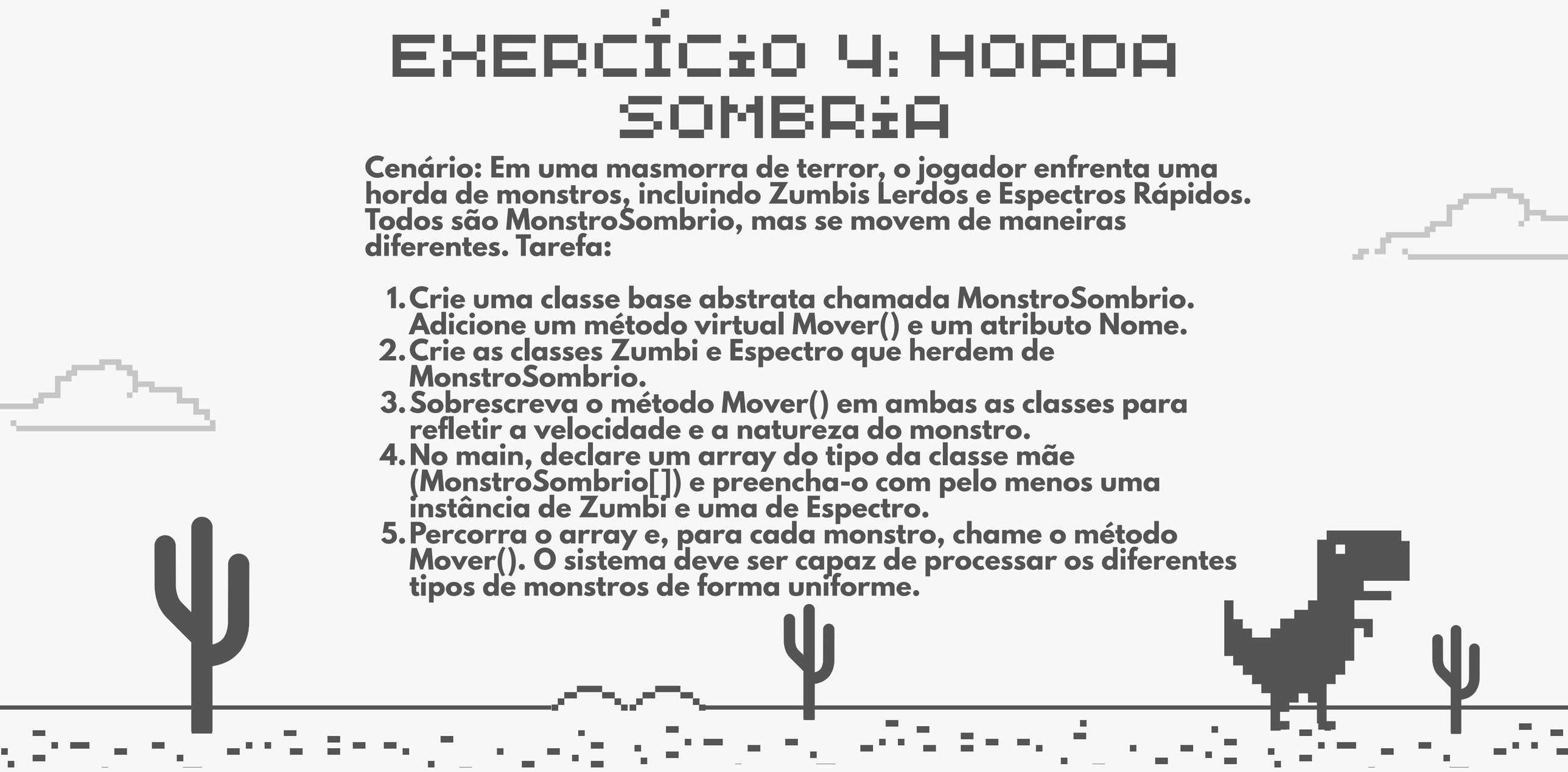
1. **Crie a classe Feitico (Propriedades).**
2. **Crie a classe Grimorio que compõe uma lista (List<Feitico>) de feitiços (Composição). Crie um método para adicionar feitiços.**
3. **Crie a classe Ferramenta (ex: "Capacete", "Lanterna").**
4. **Crie a classe Maga (Frieren) que compõe um Grimorio e agrega uma lista (List<Ferramenta>) em seu construtor ou por meio de uma propriedade.**
5. **No main, crie uma lista de Ferramentas. Crie a instância de Maga (Frieren), inicializando-a com seu Grimório (automaticamente) e passando a lista de Ferramentas (Agregação).**
6. **Adicione feitiços ao Grimório de Frieren e imprima o nome das ferramentas que ela agregou.**



EXERCÍCIO 4: HORDA SOMBRIA

Cenário: Em uma masmorra de terror, o jogador enfrenta uma horda de monstros, incluindo Zumbis Lerdos e Espectros Rápidos. Todos são MonstroSombrio, mas se movem de maneiras diferentes. Tarefa:

1. **Crie uma classe base abstrata chamada MonstroSombrio. Adicione um método virtual Mover() e um atributo Nome.**
2. **Crie as classes Zumbi e Espectro que herdem de MonstroSombrio.**
3. **Sobrescreva o método Mover() em ambas as classes para refletir a velocidade e a natureza do monstro.**
4. **No main, declare um array do tipo da classe mãe (MonstroSombrio[]) e preencha-o com pelo menos uma instância de Zumbi e uma de Espectro.**
5. **Percorra o array e, para cada monstro, chame o método Mover(). O sistema deve ser capaz de processar os diferentes tipos de monstros de forma uniforme.**




GAME OVER