MODULO 7
Polimorfismo, Classes Abstratas, Interfaces

Considere as classes Pessoa, Aluno e Professor da aula passada e as classes a seguir para compreender o funcionamento polimórfico da classe Maratona.

Interface Corredor

```
public interface ICorredor {
	void correr();
}
```

Classe Atleta que herda de Pessoa e implementa ICorredor

```
public class Atleta extends Pessoa implements ICorredor {

	public Atleta(String nome, String sobrenome, int idade) {
		super(nome, sobrenome, idade);
	}

	@Override
	public void correr() {
		System.out.print("Eu corro com duas pernas");

	}
}
```

Classe Animal

```
public class Animal {
	private boolean domesticable;
	public void comer(){
		System.out.println("Eu estou comendo");
	}
}
```

Class Tartaruga que herda de Animal

```
public class Tartaruga extends Animal {

	@Override
	public void comer(){
		System.out.println("Eu como alface");
	}
}
```

Classe Cachorro que herda de Animal e implementa ICorredor

```
public class Cachorro extends Animal implements ICorredor{

	@Override
	public void correr() {
		System.out.print("Eu corro com 4 patas");

	}
}
```

Classe Maratona

```
public class Maratona {

	private ArrayList <ICorredor> corredores =new ArrayList<ICorredor>();

	public void inscreverCorredor(ICorredor corredor){
		corredores.add(corredor);
	}
	// Método polimórfico, cada corredor pode estar implementado de uma
maneira especifica
	public void comecarMaratona(){
		int i=1;
		for (ICorredor corredor : corredores) {
			System.out.print("\nCorredor " + i++ +":  ");
			corredor.correr();
		}
	}
	public static void main(String args[]){
		Maratona m = new Maratona();
		m.inscreverCorredor(new Cachorro());
		m.inscreverCorredor(new Atleta("Eustanio", "Da Silva", 21));
		m.inscreverCorredor(new Atleta("Eusebio", "Pereira", 21));
		m.inscreverCorredor(new Cachorro());
		m.comecarMaratona();
	}
}
```

Saída ao rodar a classe Maratona:

```
Corredor 1:  Eu corro com 4 patas
Corredor 2:  Eu corro com duas pernas
Corredor 3:  Eu corro com duas pernas
Corredor 4:  Eu corro com 4 patas
```

Lista de Exercícios 7

1. Modifique a implementação do método comer da classe Animal para indicar que é um método abstrato.
   Mude também a declaração da classe Animal para se adaptar à mudança.
   Finalmente, como o método é abstrato, deverá implementar comer nas classes derivadas da classe animal com uma implementação que imprima os alimentos que cada animal come.

2. Crie uma classe TesteAnimal que possua um método main.
   Dentro do main, crie 5 animais (cachorros e tartarugas) e adicione-os a uma coleção chamada lstAnimais (pode ser do tipo ArrayList).
   Finalmente adicione o código a seguir ao seu método main.

```
System.out.print("Imprimindo os alimentos que comen todos os animais: ");
for (Animal ani: lstAnimais) {
      ani.comer();
}
```

3. Adicione uma Classe Tigre que herde de Animal e implemente Corredor e inscriva-o na maratona.
   Deverá implementar o método correr do Tigre obrigatoriamente pois ele está implementando a interface corredor. A sua implementação deverá imprimir: “Eu sou o mais veloz corredor de 4 patas”.
   Complete a classe Tigre respeitando os métodos abstratos que herdou.
   Finalmente teste a saida da classe maratona.

4. Adicione duas Classes que herdem de Tartaruga: TartarugaComun e TartarugaMarinha.
   Especifique o método comer da classe Tartaruga como abstract e faça as mudanças necessárias nas classes envolvidas: Tartaruga, TartarugaComun, TartarugaMarina e TesteAnimais para conservar essa mudança.
   No método comer deverá imprimir alimentos específicos da TartarugaComun (“Eu como vegetais”) e da TartarugaMarinha (“Eu como medusas”)
   Finalmente rode novamente a classe TesteAnimais.