Classes Abstratas e Interfaces

Interfaces em Java

- Podemos pensar que uma interface funciona como um contrato ou como um protocolo.
- Declaramos uma interface de maneira muito semelhante a uma classe, porém interfaces não podem ser instanciadas, ou seja, não podemos criar um objeto a partir de uma interface.
- Interfaces são criadas para serem utilizadas como base para implementar uma class.

Interfaces em Java

- Tomemos como ponto de partida o nosso exercício das formas.
- Não podemos garantir que todas as implementações de formas possuem os métodos calcularArea() e calcularPerimetro().
- Para forçar que todas as formas possuam estes métodos, podemos criar uma interface que deve ser utilizada pelas classes de formas.
- O uso da interface forçara a implementação destes métodos.

Classes Abstratas

- As classes abstratas são semelhantes às interfaces, porém elas podem ter alguns métodos implementados e podem definir atributos de classe.
- As interfaces só indicam as assinaturas dos métodos.
- Classes Abstratas também não podem ser instanciadas, ou seja, não podemos criar objetos a partir delas.

Classes Abstratas e Interfaces

Mas qual a vantagem de se ter um tipo o qual não se pode instanciar?

Classes Abstratas e Interfaces

- Esses tipos oferecem aos desenvolvedores informações de como uma classe, que queira ser de um tipo específico, deve ser implementado.
- No caso das classes abstratas, elas oferecem também implementações básicas que podem ser sobrescritas ou implementações finais que não podem, oferecendo assim uma implementação inicial ou um método comum entre as implementações da classe.
- Isto torna os programas mais concisos e evita que diferentes programadores implementem classes de uma mesma categoria, por assim dizer, que possuam métodos diferentes.

Exemplos

Interface para uma Forma

- Ao lado vemos a definição de uma interface, que deve ser utilizada por aqueles que querem implementar (*implements*) uma Forma.
- Primeiro substituímos a palavra-chave class, por *interface*.
- Depois listamos as assinaturas dos métodos sem uma implementação.

```
public interface Forma
{

    float calcularArea();

    float calcularPerimetro();

}
```

Classe Abstrata

- Ao lado a implementação de uma interface.

```
public class Retangulo implements Forma {
	private float base;
	private float altura;

	public float calcularArea() {
		return base*altura;
	}

	public float calcularPerimetro() {
		return 2*base + 2*altura;
	}

	public void imprimir() {
		System.out.println("A Altura é " + altura + ", a base é " + base);
	}

	public void setBaseEAltura(float b, float a) {
		base = b;
		altura = a;
	}
}
```

Classe Abstrata

- Ao lado vemos a definição de uma classe abstrata, elas devem ser herdadas (*extends*) por aqueles que desejem utilizá-las.
- Adicionamos a palavra-chave *abstract* antes de class.
- Podemos então definir tanto assinaturas como implementar métodos.

```
public abstract class Forma
{
    private String nome;

    public abstract float calcularArea();

    public abstract float calcularPerimetro();

    public void setNome(String nome) {
        this.nome = nome;
    }

    public void imprimir() {
        System.out.println("Imprimindo a partir da classe Abstrata.");
        System.out.println(this.nome + " não sobrescreve método imprimir.");
    }
}
```

Classe Abstrata

- Ao lado a implementação de uma classe utilizando uma classe abstrata como base.

```
public class Retangulo extends Forma {
        private float base;
        private float altura;

        public float calcularArea() {
                return base*altura;
        }

        public float calcularPerimetro() {
                return 2*base + 2*altura;
        }

        public void setBaseEAltura(float b, float a) {
                base = b;
                altura = a;
        }
}
```

Exercício

- Expanda o exercício do Jogo, identifique classes que potencialmente poderiam ter uma Classe Abstrata ou uma Interface como base, implemente-as e as utilize nas classes.
  - Dica: Sempre que tiver uma classe com métodos, e que essa classe dificilmente ou nunca será instanciada diretamente ela é uma candidata a ser Abstrata ou uma Interface.