

6,0 + 4,0

1. Observe o código abaixo e responda as questões a seguir.

```
abstract class ItemBiblioteca {
    protected String titulo;
    protected String autor;
    protected boolean emprestado;

    public ItemBiblioteca(String titulo, String autor) {
        this.titulo = titulo;
        this.autor = autor;
        this.emprestado = false;
    }
    public abstract String getDescricao();
    public boolean isEmprestado() {
        return emprestado;
    }
}

interface Emprestavel {
    public void emprestar();
    public void devolver();
}

class Revista extends ItemBiblioteca {
    private String editora;

    public Revista(String titulo, String autor, String editora) {
        super(titulo, autor);
        this.editora = editora;
    }

    @Override
    public String getDescricao() {
        return "Revista: " + titulo + " por " + autor + " editora " + editora;
    }
}

class Livro extends ItemBiblioteca {
    private int edicao;

    public Livro(String titulo, String autor, int edicao) {
        super(titulo, autor);
        this.edicao = edicao;
    }

    @Override
    public String getDescricao() {
        return "Livro: " + titulo + " por " + autor + " por " + edicao;
    }
}

class ItemAdapter implements Emprestavel {
    private ItemBiblioteca item;

    public ItemAdapter(ItemBiblioteca item) {
        this.item = item;
    }

    @Override
    public void emprestar() {
        if(item.isEmprestado()) System.err.println("O item já encontra-se emprestado.");
        else {
            item.emprestado = true;
            System.out.println("Empréstimo realizado com sucesso.");
        }
    }

    @Override
    public void devolver() {
        if(!item.isEmprestado()) System.err.println("O item não encontrava-se emprestado.");
        else {
            item.emprestado = false;
            System.out.println("Devolução realizado com sucesso.");
        }
    }
}
```

- 10 a. Explique o conceito de herança na orientação a objetos. Quais são as vantagens e desvantagens de usar herança? Identifique onde está sendo utilizado o conceito de herança no exemplo dado.
- 10 b. O que é polimorfismo em orientação a objetos? Onde o polimorfismo está sendo utilizado no exemplo acima? Como o polimorfismo contribui para a flexibilidade do código?
- 10 c. Descreva como os conceitos de interface e classe abstrata estão sendo utilizados no exemplo. Quando você usaria uma interface em vez de uma classe abstrata e vice-versa?
- 10 d. Qual a finalidade de uma classe adaptadora? Explique a função da classe adaptadora no exemplo da questão.
- 20 e. Analise o código a seguir em conjunto com o exemplo anterior, aponte as linhas que contem erros e comente qual o problema encontrado.

```
1. public class Biblioteca {
2.     public static void main(String[] args) {
3.         Livro livro = new Livro("O Senhor dos Anéis", "Tolkien", 3);
4.         livro.emprestar();

5.         Revista revista = new Revista("National Geographic", "Vários", "The Walt Disney Company");
6.         System.out.println(revista.getDescricao());

7.         ItemAdapter livroAdaptado = new ItemAdapter(new Livro("O Senhor dos Anéis", "Tolkien", 3));
8.         livroAdaptado.emprestar();

9.         ItemAdapter revistaAdaptada = new ItemAdapter(revista);
10.        System.out.println(revistaAdaptada.getDescricao());

11.        revistaAdaptada.devolver();
12.    }
13. }
```


a) *Franga* na orientação a objetos é o relacionamento entre uma classe "mã" e outra "filha". Na qual a classe filha herdada todas atribuições, métodos e estados da classe mãe.

Dentre os vantagens da *Franga*, está a utilização de código, bem como a capacidade de expandir os projetos e facilitar a sua manutenção.

Na *UML*, a *casca* de *Franga* é criada pelas classes "Pavista" e "Linha", que herdam de "ItemBibliotecas".

b) *Polimorfismo* é a capacidade de um objeto em se comportar como objeto de diferentes classes, e é usada na classe *ItemAdopter* no código acima, onde qual classe recebe um objeto "ItemBibliotecas" (ou seus descendentes) e vai incorporar nos seus atributos e métodos dessa forma.

A *polimorfismo* contribui na flexibilidade do código na medida que permite a atribuição de um objeto de uma classe filha ao de uma classe mãe, permitindo maior liberdade de código.

c) Na *UML*, *Exportável* é uma interface, implementada por uma classe adaptadora; e *ItemBibliotecas* é uma classe abstrata, implementada por suas filhas. Uma classe abstrata tem melhor vida quando quer-se criar mais classes com parâmetros e métodos semelhantes, evitando repetição de código.

Uma interface seria melhor usada quando se tem classes com traço único (nome e parâmetros) mas que podem ter diferentes implementações, a depender da classe que as implementa.

d) Uma classe adaptadora serve para conectar interface com diferentes implementações, evitando que se criem várias interfaces/implementações sem necessidade. Na *UML*, a classe adaptadora "ItemAdopter" implementa a interface *Exportável* e conecta com a classe "ItemBibliotecas" tornando possível que os métodos interajam com os objetos da biblioteca adequadamente.

l) na linha 9, a música Impetor não pertence a classe lírica ou Itimbilística, mas sim a classe Itimodótila, tornando-se Gamada ilm lra.

linha 10, a música gtilicrista (1); pertence a classe reposita, logo, pode ser impressa na tela pela polimorfismo da classe reposita.

Na linha 11, o item não foi impresso, como a função de derivar de como redigido.