Respostas

Questão 1

- Uma classe em Java é o que define um tipo de objeto, ou seja, representa a estrutura de dados (atributos, construtores, métodos…).
- Um objeto é uma instância criada a partir de uma determinada classe. A classe AlunoTeste representa o objeto; dentro dela, é preciso chamar o método main. Após instanciar o objeto, é possível declarar valores aos atributos e chamar os métodos.

A diferença é que uma classe define um objeto, assim gerando sua estrutura, e já o objeto depende de uma classe para ser instanciado, e posteriormente declara os atributos e chama os métodos.

Exemplos

```
Em C++

1. Classe e Objeto Representando um Carro:
Cpp (p.)
// Classe em C++
class Carro {
public:
string modelo;
string marca;
int ano;
};
// Objeto em C++
Carro pesqCar;
pesqCar.modelo = “SUV”;
pesqCar.marca = “Vw T-Cross”;
pesqCar.ano = 2023;
2. Classe e Objeto Representando um Livro:
Cpp (p.)
// Classe em C++
class Livro {
public:
string titulo;
string autor;
};
// Objeto em C++
Livro descLivro;
descLivro.titulo = "O homem Mais Rico da Babilônia";
descLivro.autor = "George S. Clason";

3. Classe e Objeto Representando um Cachorro:
Cpp (p.)
// Classe em C++
class Cachorro {
public:
string nome;
String raca;
string cor;
int idade;
};

// Objeto em C++
Cachorro descCachorro;
descCachorro.nome = "Thor";
descCachorro.raca = "Sem Raça definida";
```

```
descCachorro.cor = “Preto;
descCachorro.idade = “2”;

Em Java
1. Classe e Objeto Representando um Carro:

// Classe em Java
public class Carro {
string modelo;
string marca;
int ano;
}
// Objeto em Java
Carro carro1 = new Carro();
carro1.modelo = “SUV”;
carro1.marca = “Hiundai Creta”;
carro1.ano = 2023;
2. Classe e Objeto Representando um Livro:

// Classe em Java
public class Livro {
String titulo;
String autor;
}

// Objeto em Java
Livro livro1 = new Livro();
livro1.titulo = "Clean Code";
livro1.autor = "Robert C. Martin";

3. Classe e Objeto Representando um Cachorro:

// Classe em C++
public class Cachorro {
String nome;
String raca;
String cor;
int idade;
}
// Objeto em Java
Cachorro cachorro1 = new Cachorro();
cachorro1.nome = "Luz";
cachorro1.raca = "PitBull";
cachorro1.cor = “Preto e Branco;
cachorro1.idade = “3”;
}
```

Questão 2

- Declaração de Variáveis

Sintaxe:

```
tipo nomeAtribido; (SEM ATRIBUIR VALOR

tipo nomeAtribido = valor; (ATRIBUINDO VALOR)
```

Exemplos de declaração e atribuição de variáveis :

```
boolean VouF = true;
char letra = 'F';
int valor = 4500;
double x = 3.14;
```

Exemplos de declaração e sem atribuição de variáveis :

```
boolean VouF;
char letra;
int valor;
double x;
```

Tipos de dados primitivos mais comuns em Java

		Valores possíveis				
Tipos	***Primitivo***	***Menor***	***Maior***	***Valor Padrão***	***Tamanho***	***Exemplo***
Inteiro	byte	-128	127	0	8 bits	byte ex1 = (byte)1;
	short	-32768	32767	0	16 bits	short ex2 = (short)1;
	int	-2.147.483.648	2.147.483.647	0	32 bits	int ex3 = 1;
	long	-9.223.372.036.854.770.000	9.223.372.036.854.770.000	0	64 bits	long ex4 = 1l;
Ponto Flutuante	float	-1,4024E-37	3.40282347E + 38	0	32 bits	float ex5 = 5.50f;

	double	-4,94E-307	1.79769313486231570E + 308	0	64 bits	double ex6 = 10.20d;
						ou
						double ex6 = 10.20;

Caractere	char	0	65535	\0	16 bits	char ex7 = 194; ou char ex8 = 'a';
Booleano	boolean	false	true	false	1 bit	boolean ex9 = true;

C++ e Java são duas linguagens de programação amplamente utilizadas, e embora compartilhem algumas semelhanças, também têm diferenças distintas. Vamos abordar a forma de declaração de variáveis e alguns tipos de dados comuns em ambas as linguagens.

Forma de Declaração de Variáveis:

C++

```
// Declaração de variável em C++
int idade;    // declaração de uma variável inteira chamada "idade"
```

```
float altura; // declaração de uma variável de ponto flutuante chamada "altura"
char letra;   // declaração de uma variável caractere chamada "letra"

// Inicialização
idade = 25;
altura = 1.75;
letra = 'A';
Forma de Declaração de Variáveis:
```

C++:

```
cpp Copy code

// Declaração de variável em C++

int idade;

// declaração de uma variável inteira chamada "idade"

float altura;

// declaração de uma variável de ponto flutuante chamada "altura"

char letra;

// declaração de uma variável caractere chamada "letra"

// Inicialização

idade = 25;

altura = 1.75;

letra = 'A';
```

Java:

```
java

// Declaração de variável em Java

int idade;

// declaração de uma variável inteira chamada "idade"

float altura;

// declaração de uma variável de ponto flutuante chamada "altura"

char letra;

 // declaração de uma variável caractere chamada "letra"

// Inicialização

idade = 25;

altura = 1.75f;

// em Java, é necessário adicionar o sufixo 'f' para literais de ponto flutuante

letra = 'A';
```

Tipos de Dados Mais Utilizados:

C++

- **int:** Números inteiros.
- **float:** Números de ponto flutuante.
- **char:** Caracteres individuais.
- **double:** Números de ponto flutuante com maior precisão.

Java

- **int:** Números inteiros.
- **float:** Números de ponto flutuante.
- **char:** Caracteres individuais.
- **double:** Números de ponto flutuante com maior precisão.

Se observarmos as tabelas de ambas, as mesmas suportam os mesmos tipos de dados. Entretanto, há alguns pontos diferentes, como no caso da necessidade de adicionar o sufixo "f" para declarações das variáveis que utilizam o tipo ponto flutuante float em Java.

Tipo de Dado	Bits	Faixa de Valores
char	8	-128 a 127
bool	8	true ou false
int	32	-2.147.483.647 a 2.147.483.647
float	32	7 dígitos significativos
double	64	15 dígitos significativos

Questão 3

Herança **é a capacidade de uma subclasse de ter acesso as propriedades da superclasse(também chamada classe base) relacionada a está subclasse**. Com isso os atributos e métodos de uma classe são propagados de cima para baixo em um diagrama de classes.

Obs.: Estes exemplos a baixo, representa os conceitos de herança e subclasses em Java e C++, entretanto vale ressaltar que o conceito de herança múltipla em C++ não é permitida em Java. Além disso, Java usa a palavra-chave `abstract` para definir classes abstratas, enquanto C++ utiliza métodos puramente virtuais (`virtual void funcao() = 0;`).

Exemplo 1: Herança Simples

Java:

```
java

class Animal {
    void fazerSom() {
        System.out.println("Alguns sons genéricos de animal");
```

```
    }
}

class Cachorro extends Animal {
    void latir() {
        System.out.println("Au au!");
    }
}
```

C++:

```
cpp

#include <iostream>

class Animal {
public:
    void fazerSom() {
        std::cout << "Alguns sons genéricos de animal" << std::endl;
    }
};

class Cachorro : public Animal {
public:
    void latir() {
        std::cout << "Au au!" << std::endl;
    }
}
```

Exemplo 2: Herança com Construtores

Java:

```
java

class Pessoa {
    String nome;

    Pessoa(String nome) {
        this.nome = nome;
    }
}

class Estudante extends Pessoa {
    int matricula;

    Estudante(String nome, int matricula) {
        super(nome);
        this.matricula = matricula;
    }
}
```

C++:

```
cpp

#include <iostream>
#include <string>

class Pessoa {
public:
    std::string nome;
```

```
    Pessoa(std::string nome) : nome(nome) {}
};

class Estudante : public Pessoa {
public:
    int matricula;

    Estudante(std::string nome, int matricula) : Pessoa(nome),
matricula(matricula) {}
}
```

Exemplo 3: Herança e Polimorfismo

Java:

```
java

abstract class Forma {
    abstract void desenhar();
}

class Circulo extends Forma {
    void desenhar() {
        System.out.println("Desenhando um círculo");
    }
}

class Quadrado extends Forma {
    void desenhar() {
        System.out.println("Desenhando um quadrado");
    }
}
```

C++:

```
cpp

#include <iostream>

class Forma {
public:
    virtual void desenhar() = 0;
};

class Circulo : public Forma {
public:
    void desenhar() override {
        std::cout << "Desenhando um círculo" << std::endl;
    }
};

class Quadrado : public Forma {
public:
    void desenhar() override {
        std::cout << "Desenhando um quadrado" << std::endl;
    }
}
```

Exemplo 4: Herança Múltipla (C++)

C++:

```
cpp

#include <iostream>

class Animal {
public:
    void fazerSom() {
        std::cout << "Alguns sons genéricos de animal" << std::endl;
    }
};

class Voador {
public:
    void voar() {
        std::cout << "Voando" << std::endl;
    }
};

class Pato : public Animal, public Voador {
public:
    void quack() {
        std::cout << "Quack!" << std::endl;
    }
}
```

Exemplo 5: Hierarquia de Herança

Java:

```
java

class Veiculo {
    void mover() {
        System.out.println("Movendo-se");
    }
}

class Carro extends Veiculo {
    void acelerar() {
        System.out.println("Acelerando o carro");
    }
}

class CarroEsportivo extends Carro {
    void turbo() {
        System.out.println("Ativando o turbo");
    }
}
```

C++:

```
cpp

#include <iostream>

class Veiculo {
public:
    void mover() {
```

```
        std::cout << "Movendo-se" << std::endl;
    }
};

class Carro : public Veiculo {
public:
    void acelerar() {
        std::cout << "Acelerando o carro" << std::endl;
    }
};

class CarroEsportivo : public Carro {
public:
    void turbo() {
        std::cout << "Ativando o turbo" << std::endl;
    }
}
```

Questão 4

Discordo, pois Java declara suas variáveis criando uma referência a um objeto, algo que é realizado automaticamente pelo sistema que realiza a alocação de memória e desalocação, e que são tratados automaticamente pelo coletor de lixo (garbage collector). A referência em Java não aponta diretamente para uma posição de memória, mas para o objeto gerenciado pelo sistema.

Já em C++, declaramos uma de variáveis de duas maneiras: Declarando diretamente variável ou usando um ponteiro para alocar dinamicamente a memória, algo que é realizado pelo programador. É bom ressaltar que em C++ tem mais controle na manipulação de ponteiros e no gerenciamento de memória, o que pode ser poderoso, mas também requer uma atenção especial para evitar vazamentos de memória e comportamentos indesejados.