1-
A motivação para o uso da orientação a objetos sobre programação procedural é pelo fato da orientação a objetos se propôr a se aproximar mais da vida real. Deve-se codificar de uma maneira que seja mais compreensível ao nível da vida real, e não ao nível de código. Criam-se novos tipos que são representações de objetos que existem na vida real, o que pode inicialmente complicar as coisas ao nível de programação, mas ao longo prazo, e em sistemas grandes, é mais vantajoso utilizar a orientação a objetos para compreensão do que está acontecendo.

2-
Encapsulamento: O encapsulamento propõe que as características e ações de um objeto se mantenham específicos, ou seja, manter o funcionamento dos métodos de uma classe apenas para aquilo que ele se propõe. Não é necessário que se saiba os detalhes dos métodos criados. Por exemplo, para uma classe Carro, há várias características que são encapsuladas que não são necessárias que se saibam, como quais peças são utilizadas para montar o carro, de qual marca são, etc. Herança pode ser definida em uma (ou mais de uma, visto que existe também a herança múltipla) classe mãe e uma (ou mais, pois uma classe pode ser base para várias outras) classe filha. A classe mãe possui atributos e métodos que são em comum entre os seus filhos, por exemplo: Classe Veículo. A classe Veículo pode ter métodos como Acelerar(), logo o seu filho, como por exemplo a classe Carro, herdará essa "característica" da sua mãe. Polimorfismo é quando uma instância de um classe pode ter mais de uma forma. Tomando como exemplo a classe Veículo de novo, a partir dessa classe pode-se criar a classe Carro, mas também a classe Moto, visto que as duas classes são veículos, mas possuem características que as distinguem.

3-
Uma variável de instância pertence ao objeto que é criado, e normalmente possui um valor diferente para cada objeto. Uma variável de classe é uma variável que possui o valor comum a todos os objetos que são membros da classe.

4-
Os atributos em Orientação a Objetos são as características de uma classe. Uma classe pode ter membros que, que são atributos que são comuns em todas as instâncias dessa classe. Um objeto "Sedan", que é instância da classe Carro, vai possuir os atributos: Cor, Marca, etc. Da mesma maneira o objeto "Fiesta", que também é uma instância dessa classe, vai possuir os mesmos atributos, com valores diferentes.

5-
Java utiliza o padrão de camelCase, onde a primeira palavra da classe/método/atributo começa com a letra minúscula, e a segunda palavra em diante com a letra maiúscula, ex: getNome().

6-
JDK é o kit de desenvolvimento do Java, que permitem criar sistemas em Java e é composto por compilador e bibliotecas. JRE é o Ambiente de Tempo de Execução Java, que é utilizado para executar as aplicações Java e é composto por bibliotecas e pela Máquina Virtual Java. A variável JAVA_HOME é a variável de ambiente onde está instalado o JDK. No linux, ela pode ser definida configurando o arquivo de profile do java com o caminho

desejado. A mesma maneira para o windows, se deve configuar as variáveis de ambiente, selecionando o caminho desejado.

7-

```java
public class HelloWorld{
    public static void main(String[]args){
        System.out.println("Hello World!");
    }
}
```

8-

```java
public class Argumentos {

    Run | Debug
    public static void main(String[] args) {
        String sArgumentos = String.valueOf(args[0]) + String.valueOf(args[1]);
        System.out.println("Argumentos inseridos: " + sArgumentos);
    }
}
```

```
PROBLEMAS   SAÍDA   TERMINAL

PS C:\Users\josef\OneDrive\Área de Trabalho\POO\Topico 1>  & 'C:\Program Files\Eclipse Adoptium\jdk-17.0.3.7-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\josef\AppData\Roaming\Code\User\workspaceStorage\4cd55dd8ba060d0217cc2f2e0921a998\redhat.java\jdt_ws\Topico 1_afd7b456\bin' 'Argumentos' 5 10
Argumentos inseridos: 510
PS C:\Users\josef\OneDrive\Área de Trabalho\POO\Topico 1>
```

9-

```java
public class Argumentos {

    Run | Debug
    public static void main(String[] args) {
        int iArgumento = Integer.parseInt(args[0]);

        for (int idx = 1; idx <= iArgumento; ++idx) {
            System.out.print(idx + " ");
        }
    }
}
```

10-

```java
import java.util.Scanner;

public class Argumentos {

    Run | Debug
    public static void main(String[] args) {
        Scanner teclado = new Scanner(System.in);
        System.out.println(x: "Digite um numero para a contagem: ");

        int iNumero = Integer.parseInt(teclado.nextLine());

        for (int idx = 1; idx <= iNumero; ++idx) {
            System.out.print(idx + " ");
        }

        teclado.close();
    }
}
```

10- (Java Básico)

a)

```java
Run | Debug
public static void main(String[] args) {
    byte tipoByte = 127;
    short tipoShort = 32767;
    int iTipoInt = 12;
    long lTipoLong = 14;
    boolean bTipoBool = true;
    char tipoChar = 'C';
    float fTipoFloat = 2.6f;
    double dTipoDouble = 3.59;
}
```

b)
Um cast é uma conversão implícita de um tipo de dados para outro.

```java
int i = 10;
long l = (long)i;
```

c)

```java
int i = 10;
long l = (long)i;

if (l == 10) {
    System.out.print(s: "l = 10");
} else {
    System.out.print(s: "l != 10");
}
```

d)
String é uma cadeia de caracteres, e possui várias funções úteis, tais como: charAt, codePointAt, compareTo e compareToIgnoreCase, entre outros.

f)
While, do while e for. Podemos controlar os loops com o comando “break”, “continue” e através da condição de validade estabelecida nos parenteses.

g)

Argumentos.java > ...

```java
public class Argumentos {

    Run | Debug
    public static void main(String[] args) {
        for (int idx = 2; idx <= 100; ++idx) {
            if (numeroPrimo(idx))
                System.out.println(idx + " e primo");
        }
    }

    private static boolean numeroPrimo(int idx) {
        for (int j = 2; j < idx; ++j) {
            if (idx % j == 0)
                return false;
        }

        return true;
    }
}
```

h)
Será impresso o valor de c que não foi alterado, então vai ser impresso 0.

8-

```java
import java.util.Scanner;

public class Argumentos {

    Run | Debug
    public static void main(String[] args) {
        Scanner teclado = new Scanner(System.in);
        System.out.println(x: "Digite um numero para a contagem: ");

        int iNumero = Integer.parseInt(teclado.nextLine());

        for (int idx = 1; idx <= iNumero; ++idx) {
            System.out.print(idx + " ");
        }

        teclado.close();
    }
}
```

9)

```java
import java.io.File;

public class Argumentos {

    Run | Debug
    public static void main(String[] args) {
        File diretorio = new File(pathname: ".");
        String[] aArquivos = diretorio.list();

        for (String sArquivo : aArquivos) {
            System.out.println(sArquivo);
        }
    }
}
```

10-

Os tipos primitivos são boolean, byte, char, short, int, long, float e double. Uma variável do tipo primitivo pode armazenar exatamente um valor de seu tipo declarado por vez, quando outro valor for atribuído a essa variável, seu valor inicial será substituído.

```java
Run | Debug
public static void main(String[] args) {
    byte tipoByte = 127;
    short tipoShort = 32767;
    int iTipoInt = 12;
    long lTipoLong = 14;
    boolean bTipoBool = true;
    char tipoChar = 'C';
    float fTipoFloat = 2.6f;
    double dTipoDouble = 3.59;
}
```

11-

Argumentos.java > ...

```java
import java.util.Scanner;

public class Argumentos {

    Run | Debug
    public static void main(String[] args) {
        Scanner teclado = new Scanner(System.in);
        System.out.println(x: "Digite um numero: ");

        int iFatorial = Integer.parseInt(teclado.nextLine());
        int iResultado = 1;

        for (int idx = iFatorial; idx > 1; --idx) {
            iResultado *= idx;
        }

        System.out.println("Fatorial de " + iFatorial + " = " + iResultado);
        teclado.close();
    }
}
```

b) Com o 10 e o 30 o programa ainda imprime os resultados, porém com o 40 já não há retorno válido, pois o resultado do fatorial de 40 é maior do que o tipo de dado int pode aguentar. Alterando o código para utilizar BigInt, que suporta números maiores, o código volta a funcionar.

```java
public class Argumentos {

    Run | Debug
    public static void main(String[] args) {
        Scanner teclado = new Scanner(System.in);
        System.out.println(x: "Digite um numero: ");

        int iFatorial = Integer.parseInt(teclado.nextLine());
        BigInteger iResultado = BigInteger.ONE;

        for (int idx = iFatorial; idx > 1; --idx) {
            iResultado = iResultado.multiply(BigInteger.valueOf(idx));
        }

        System.out.println("Fatorial de " + iFatorial + " = " + iResultado);
        teclado.close();
    }
}
```