****

**UNIVERSIDADE FEDERAL DE GOIÁS**

**ENGENHARIA DE SOFTWARE**

**SOFTWARE CONCORRENTE E DISTRIBUIDO – INF0298**

Discente:

Adriel Lenner Vinhal Mori

Docente:

Vagner Jose Do Sacramento Rodrigues

**Lab Thread 2 – 16 jun 2023**

Goiânia

2023

**1.**

**a.**

public class Contador extends Thread {

    public void run() {

        for (int i = 0; i <= 10; i++) {

            System.out.println(i);

        }

    }

}

public class TesteContador {

    public static void main(String[] args) {

        Contador contador = new Contador();

        contador.start();

    }

}

**b.**

// Classe Contador agora implementa Runnable

public class Contador implements Runnable {

    public void run() {

        for (int i = 0; i <= 10; i++) {

            System.out.println(i);

        }

    }

}

public class TesteContador {

    public static void main(String[] args) {

        Contador contador = new Contador();

        // Para inicar a thread basta chamar pela função run() implementada em Runnable

        contador.run();

    }

}

**c.**

public class Contador implements Runnable {

    private int contadorId;

    public Contador(int id) {

        contadorId = id;

    }

    public void run() {

        for (int i = 0; i <= 10; i++) {

            System.out.println("Contador " + contadorId + ": " + i);

        }

    }

}

public class TesteContador {

    public static void main(String[] args) {

        int numThreads = 5; // Número de threads a serem criadas

        Thread[] threads = new Thread[numThreads]; // Array para armazenar as threads

        // Criando as threads

        for (int i = 0; i < numThreads; i++) {

            Contador contador = new Contador(i + 1);

            threads[i] = new Thread(contador);

        }

        // Iniciando as threads

        for (int i = 0; i < numThreads; i++) {

            threads[i].start();

        }

    }

}

**2.**

public class Produtor implements Runnable {

  private Deposito deposito;

  private int tempoProducao;

  public Produtor(Deposito dep, int tempo) {

    deposito = dep;

    tempoProducao = tempo;

  }

  public void run() {

    while (true) {

      deposito.armazenar();

      try {

        Thread.sleep(tempoProducao \* 1000);

      } catch (InterruptedException e) {

        // Tratamento de interrupção da thread

        System.out.println("Produtor interrompido.");

        break;

      }

    }

  }

}

public class Consumidor implements Runnable {

  private Deposito deposito;

  private int tempoRetirada;

  public Consumidor(Deposito dep, int tempo) {

    deposito = dep;

    tempoRetirada = tempo;

  }

  public void run() {

    while (true) {

      int caixasRetiradas = deposito.retirar();

      if (caixasRetiradas == 0) {

        System.out.println(

          "Consumidor bloqueado. Aguardando caixas para retirar..."

        );

      } else {

        System.out.println(

          "Consumidor retirou " + caixasRetiradas + " caixas."

        );

      }

      try {

        Thread.sleep(tempoRetirada \* 1000); // Converte o tempo para milissegundos

      } catch (InterruptedException e) {

        // Tratamento de interrupção da thread

        System.out.println("Consumidor interrompido.");

        break;

      }

    }

  }

}

public class Deposito {

  private int items = 0;

  private final int capacidade = 10;

  public synchronized int retirar() {

    while (items == 0) {

      try {

        wait(); // Aguarda até que haja caixas disponíveis para retirar

      } catch (InterruptedException e) {

        e.printStackTrace();

      }

    }

    items--;

    System.out.println("Caixa retirada: Sobram " + items + " caixas");

    notifyAll(); // Notifica todas as threads em espera

    return 1;

  }

  public synchronized int armazenar() {

    while (items == capacidade) {

      try {

        wait(); // Aguarda até que haja espaço disponível para armazenar

      } catch (InterruptedException e) {

        e.printStackTrace();

      }

    }

    items++;

    System.out.println("Caixa armazenada: Passaram a ser " + items + " caixas");

    notifyAll(); // Notifica todas as threads em espera

    return 1;

  }

  public static void main(String[] args) {

    Deposito dep = new Deposito();

    Produtor p = new Produtor(dep, 2);

    Consumidor c = new Consumidor(dep, 1);

    Thread produtorThread = new Thread(p);

    Thread consumidorThread = new Thread(c);

    produtorThread.start();

    consumidorThread.start();

    System.out.println("Execução do main da classe Deposito terminada!");

  }

}