53 45 52 45 49 20 46 49 45 4c 20
41 4f 53 20 50 52 45 43 45 49 54
4f 53 20 44 41 20 48 4f 4e 52 41
20 45 20 44 41 20 43 49 c3 8a 4e
43 49 41 2c 20 50 52 4f 4d 4f 56
45 4e 44 4f 20 4f 20 55 53 4f 20
45 20 4f 20 44 45 53 45 4e 56 4f
4c 56 49 4d 45 4e 54 4f 20 44 41
20 49 4e 46 4f 52 4d c3 81 54 49
43 41 20 45 4d 20 42 45 4e 45 46
c3 8d 43 49 4f 20 44 4f 20 43 49
44 41 44 c3 83 4f 20 45 20 44 41
20 53 4f 43 49 45 44 41 44 45 2e

RESIDÊNCIA DE SOFTWARE

CAPACITAR
TREINAR
EMPREGAR
TRANSFORMAR

Firjan SENAI

serratec
Parque Tecnológico Região Serrana

JAVA I

Coleções e Datas

10/08/2020

EXERCÍCIO – REVISANDO CONCEITOS DE AULA ANTERIORES - EXCEPTION

Classe Scanner

Facilita a entrada de dados no Java, surgiu a partir do Java 5 com o objetivo de facilitar a entrada de dados no modo Console.
Uma das características mais interessante da classe Scanner é a possibilidade de obter o valor digitado diretamente no formato do tipo primitivo que o usuário digitar.
Para isso basta utilizarmos os métodos next do tipo primitivo no formato nextTipoDado() conforme exemplo abaixo:

```
Scanner scanner = new Scanner(System.in);
int numeroInteiro = scanner.nextInt();
byte numeroByte = scanner.nextByte();
long numeroLong = scanner.nextLong();
boolean booleano = scanner.nextBoolean();
float numeroFloat = scanner.nextFloat();
double numeroDouble = scanner.nextDouble();
```

Exemplo:

```
public class Console {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        System.out.println("Qual o seu nome:");
        String nome= sc.next();
        System.out.println("Bom dia !!" + nome );
    }

}
```

EXERCÍCIO – REVISANDO CONCEITOS DE AULA ANTERIORES - EXCEPTION

Exercício Exceções

Criar um programa que leia um número inteiro e trate a exceção caso uma letra seja digitada. Tratar a exceção do tipo **InputMismatchException**

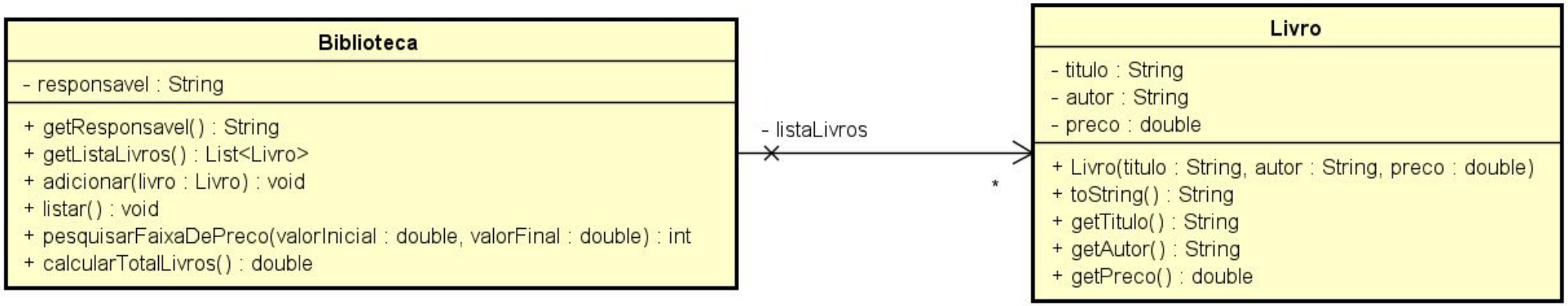
```
public class Console {

    public static void main(String[] args) {
        try {
            Scanner sc = new Scanner(System.in);
            System.out.println("Digite um número:");
            int numero = sc.nextInt();

        } catch (InputMismatchException e) {
            System.out.println("Erro !! Digite um número inteiro !!");
        }
    }
}
```

EXERCÍCIO – REVISANDO CONCEITOS DE AULA ANTERIORES

Criar a classe conforme diagrama abaixo:
A classe Biblioteca deverá ter um método para adicionar um novo livro, listar todos os livros, pesquisar por faixa de preço, calcular o total em dinheiro de livros.
Criar uma classe com o método main para fazer os testes das operações no console.



RESPOSTA

```java
public class Livro {
    private String titulo;
    private String autor;
    private double preco;

    public Livro(String titulo, String autor, double preco) {
        super();
        this.titulo = titulo;
        this.autor = autor;
        this.preco = preco;
    }

    @Override
    public String toString() {
        return "Livro [titulo=" + titulo + ", autor=" + autor +  " preco=" + preco + "]";
    }

    public String getTitulo() {
        return titulo;
    }

    public String getAutor() {
        return autor;
    }


    public double getPreco() {
        return preco;
    }

}
```

RESPOSTA

```java
public class Biblioteca {
    private String responsavel;
    private List<Livro> listaLivros = new ArrayList<Livro>();

    public String getResponsavel() {
        return responsavel;
    }

    public List<Livro> getListaLivros() {
        return listaLivros;
    }

    public void adicionar(Livro livro) {
        listaLivros.add(livro);
    }

    public void listar() {
        for (Livro livro : listaLivros) {
            System.out.println(livro.getAutor() + " "+ livro.getTitulo() + " "+ livro.getPreco());
        }
    }
    public int pesquisarFaixaDePreco(double valorInicial, double valorFinal) {
        int quantidade = 0;
        for (Livro livro : listaLivros) {
            if (livro.getPreco() >= valorInicial && livro.getPreco() <= valorFinal) {
                quantidade++;
            }
        }
        return quantidade;
    }

    public double calcularTotalLivros() {
        double total = 0;
        for (Livro livro : listaLivros) {
            total += livro.getPreco();
        }
        return total;
    }
}
```

RESPOSTA

```java
public class TesteBiblioteca {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Scanner entradaString = new Scanner(System.in);
        int menu;
        double preco, valorInicial, valorFinal;
        Livro livro;
        String titulo, autor;
        Biblioteca biblioteca = new Biblioteca();
        do {
            exibirMenu();
            menu = sc.nextInt();
            switch (menu) {
            case 1:
                System.out.print("Titulo:");
                titulo = entradaString.nextLine();
                System.out.print("Autor:");
                autor = entradaString.nextLine();
                System.out.print("Preço:");
                preco = sc.nextDouble();

                livro = new Livro(titulo, autor, preco);
                biblioteca.adicionar(livro);
                break;
            case 2:
                System.out.println("\n========Listagem de Livros============");
                biblioteca.listar();
                System.out.println("=====================================");
                break;
            case 3:
                System.out.println("========Pesquisa por faixa de preço======");
                System.out.print("Digite o valor inicial:");
                valorInicial = sc.nextDouble();
                System.out.print("Digite o valor final:");
                valorFinal = sc.nextDouble();
                System.out.println("Exitem" + biblioteca.pesquisarFaixaDePreco(valorInicial, valorFinal) + " livros");
                break;
            case 4:
                System.out.println("Total de livros:" + biblioteca.calcularTotalLivros());

                break;
            case 5:
                System.out.println("Saindo do sistema");
                break;

            default:
                System.out.println("Opção inválida");
                break;
            }
        } while (menu != 5);

    }

                System.out.println("Opção inválida");
                break;
            }
        } while (menu != 5);

    }

    public static void exibirMenu() {
        System.out.println("*******BIBLIOTECA DA FACULDADE********");
        System.out.println("1-Adicionar");
        System.out.println("2-Listar");
        System.out.println("3-Pesquisar por Preço");
        System.out.println("4-Calcular Total");
        System.out.println("5-Sair");
        System.out.println("Digite a opção:");
    }

}
```

MAPAS - JAVA.UTIL.MAP

- Um mapa é composto por um conjunto de associações entre um objeto chave a um objeto valor. É equivalente ao conceito de dicionário, usado em várias linguagens.

```java
public class Main {
    public static void main(String[] args) {
        HashMap<String, String> mapaEstados = new HashMap<>();
        mapaEstados.put("AC", "Acre");
        mapaEstados.put("AL", "Alagoas");
        mapaEstados.put("AP", "Amapá");
        mapaEstados.put("AM", "Amazonas");
        mapaEstados.put("BA", "Bahia");
        mapaEstados.put("CE", "Ceará");
        mapaEstados.put("DF", "Distrito Federal");
        mapaEstados.put("ES", "Espírito Santo");
        mapaEstados.put("GO", "Goiás");

        for (String sigla : mapaEstados.keySet()) {
            System.out.print(sigla + " ");
        }
    }
}
```

Suas principais implementações são:

- HashMap
- TreeMap
- Hashtable

Percorrendo Elementos obtendo a chave

MAPAS - JAVA.UTIL.MAP

A sintaxe deve obedecer a lugares apontados da chave e valor, pois cada chave leva somente um elemento.
Map<E> mapa = new Type();

Sintaxe:
E - é o objeto declarado, podendo ser classes Wrappers ou tipo de coleção.
Type - é o tipo de objeto da coleção a ser usado.

Classe HashMap
Os elementos não são ordenados. É rápida na busca/inserção de dados. Permite inserir valores e chaves nulas;

```
public class TesteMap {
    public static void main(String[] args) {
        Map<Integer, String> mapaNomes = new HashMap<Integer, String>();
        mapaNomes.put(1, "João");
        mapaNomes.put(2, "Maria");
        mapaNomes.put(3, "Gerson");

        mapaNomes.replace(2, "Ana");  //Modificando conteúdo
        mapaNomes.remove(3);

        System.out.println(mapaNomes);

        System.out.println(mapaNomes.get(2));
        System.out.println(mapaNomes.keySet());
        System.out.println(mapaNomes.values());
        System.out.println(mapaNomes.entrySet());

        for(int i=1; i< mapaNomes.size(); i++) {
            System.out.println(mapaNomes.get(i));
        }

    }
}
```

EXERCÍCIOS

Criar uma classe para inserir em um ´mapa de marcas e modelos de alguns veículos e percorrer para imprimir a chave o valor.

```
public class TesteVeiculo {
    public static void main(String[] args) {
        Map<String, String> carros = new HashMap<String, String>();
        carros.put("VW", "Gol");
        carros.put("Fiat", "Siena");
        carros.put("Ford", "Fiesta");
        carros.put("Renault", "Sandero");

        // Obtendo a chave - Marcas dos carros
        for (String s : carros.keySet()) {
            System.out.println(s);
        }

        // Obtendo a chave e o valor - Marca e modelo dos carros
        for (Map.Entry<String, String> entrada : carros.entrySet()) {
            System.out.println(entrada);
        }
    }
}
```

TRABALHANDO COM DATAS NO JAVA

Classe Date

A data representa o tempo, um tempo é composto por ano, mês, dia atual, minuto atual, entre outros atributos e métodos que essa classe possui. Hoje a maioria dos métodos da classe Date estão classificados como deprecated, métodos que não são mais utilizados. A classe Date foi substituída pela Calendar.

```
public class Datas {

    public static void main(String[] args) {

            Date dataDeHoje = new Date();

            System.out.println("Data de Hoje:" + dataDeHoje);
            System.out.println("Milisegundos desde 1 janeiro de 1970:" + dataDeHoje.getTime());
            System.out.println("Dia de Hoje" + dataDeHoje.getDate());

            SimpleDateFormat formato = new SimpleDateFormat("dd/MM/yyyy");
            String dataFormatada = formato.format(dataDeHoje);
            System.out.println("Data Formatada:" + dataFormatada);

    }

}
```

TRABALHANDO COM DATAS NO JAVA

Classe Calendar

É uma classe abstrata que não pode ser instanciada, portanto para obter um calendário é necessário usar o método estático.
A classe produz valores de todos os campos de calendário necessários para implementar a formatação de data e hora, para uma determinada língua e estilo de calendário.

```
public class Calendars {

    public static void main(String[] args) {
        Calendar hoje = Calendar.getInstance();
        System.out.println(hoje);

        int ano = hoje.get(Calendar.YEAR);
        int mes = hoje.get(Calendar.MONTH);
        int dia = hoje.get(Calendar.DAY_OF_MONTH);
        int hora = hoje.get(Calendar.HOUR_OF_DAY);
        int minutos = hoje.get(Calendar.MINUTE);
        int segundos = hoje.get(Calendar.SECOND);

        System.out.println(dia + "-" + mes + "-" + ano);
        System.out.println(hora);

        //mês em Calendar começa com 0
        System.out.printf("Hoje é: %02d/%02d/%d",dia,mes+1,ano);
        System.out.printf("\nHora: %d:%d:%02d",hora,minutos,segundos);
    }

}
```

TRABALHANDO COM DATAS NO JAVA

Classe LocalDate

A manipulação de datas usando **Date** e **Calendar** para criação de filtros, cálculos e conversões eram trabalhosas nessas classes.
E também escrevíamos muito código. A partir do Java 8 foi disponibilizada uma nova API de data e hora. O **LocalDate** foi introduzido para facilitar o trabalha com datas. Ele fica no pacote **java.time**

```
public class TesteLocalDate {

    public static void main(String[] args) {
        LocalDate hoje = LocalDate.now();
        System.out.println(hoje);

        System.out.println(LocalDate.of(2020, 8, 10));

        System.out.println(LocalDate.parse("2020-08-10"));

        //ADICIONANDO 30 DIAS
        System.out.println(hoje.plusDays(30));
        //8 DIAS ATRÁS
        System.out.println(hoje.minusDays(8));
        //DOIS MESES ATRÁS
        System.out.println(hoje.minus(2, ChronoUnit.MONTHS));
        System.out.println(hoje.getDayOfWeek() + " " + hoje.getDayOfMonth() +" " + hoje.getDayOfYear());
        //SE É ANO BISEXTO
        System.out.println(hoje.isLeapYear());

    }

}
```

TRABALHANDO COM DATAS NO JAVA

Trabalhando com comparações de Datas

Frequentemente quando trabalhamos com datas necessitamos fazer operações de comparação, diferença entre datas, etc. Para isso, essa nova API veio com outras facilidade para que possamos fazer essas operações de maneira mais simples.

- isAfter
- isBefore
- isEquals
- LocalDate.now()
- plusDays()
- minusYears()

Uma classe que também nos ajuda nessa comparação é a **Period**. Ela representa uma quantidade de tempo em anos, meses e dias

```
Period periodo = Period.between(dataInicio, dataFim);
periodo.getYears()
```


Cálculo de anos entre duas datas

EXERCÍCIO

1. Escreva um programa que escreva na console o dia de hoje, o dia da semana, mês e ano atual.
2. Alem disso, exiba a quantidade de anos desde sua data de nascimento.

```java
public class MainLocalDate {
    public static void main(String[] args) {

        LocalDate localDate = LocalDate.now().minusDays(1);

        System.out.println(localDate);
        System.out.println("Dia da semana: " + localDate.getDayOfWeek().name());
        System.out.println("Dia da semana: " + localDate.getDayOfWeek().getValue());
        System.out.println("Mes: " + localDate.getMonthValue());
        System.out.println("Mes: " + localDate.getMonth().name());
        System.out.println("Ano: " + localDate.getYear());

        LocalDate dataAniversario = LocalDate.of(2015, 3, 5);
        Period periodo = Period.between(dataAniversario, localDate);
        System.out.println(periodo.getYears());
    }
}
```

TRABALHANDO COM HORAS NO JAVA

Classe LocalTime

Função para manipulação de horas.

```
public class TesteLocalTime {

    public static void main(String[] args) {
        LocalTime hora = LocalTime.now();
        System.out.println(hora);

        System.out.println(LocalTime.of(20, 10));
        System.out.println(LocalTime.parse("20:10"));

        System.out.println("Hora Atual + 60 minutos: "+hora.plusMinutes(60));
        System.out.println("Hora Atual - 10 minutos: " + hora.minusMinutes(10));


    }

}
```

TRABALHANDO COM DATAS E HORAS NO JAVA

Classe LocalDateTime

Função para manipulação de datas e horas.

```
public class TesteLocalDateTime {

    public static void main(String[] args) {
        LocalDateTime dataHora= LocalDateTime.now();
        System.out.println(dataHora);

        System.out.println(dataHora.plusDays(10));

        System.out.println(LocalDateTime.of(2020,8,10,13,00,00));

        //Exibe fuso padrão do sistema
        ZoneId fuso = ZoneId.systemDefault();
        System.out.println(fuso);

        //Atribui um fuso
        fuso = ZoneId.of("America/Sao_Paulo");

        //Exibe todos os fusos disponíveis
        Set<String>fusos = ZoneId.getAvailableZoneIds();
        for(String f: fusos) {
            System.out.println(f);
        }
    }
```

TRABALHANDO COM DATAS E HORAS NO JAVA

Duration de tempo entre dois objetos LocalDateTime

Podemos utilizar a classe **Duration** para nos auxiliar no calculo da duração entre dois objetos **LocalDateTime**

```
public class MainLocalDateTime {
    public static void main(String[] args) {
        LocalDateTime oldDate = LocalDateTime.of(2016, Month.AUGUST, 31, 10, 20, 55);
        LocalDateTime newDate = LocalDateTime.of(2016, Month.DECEMBER, 9, 10, 21, 55);

        System.out.println(oldDate);
        System.out.println(newDate);

        Duration duracao = Duration.between(oldDate, newDate);
        System.out.println(duracao.getSeconds());
    }
}
```

TRABALHANDO COM DATAS E HORAS NO JAVA

Formatando datas e horas

A formatação de data para diferentes padrões também ficou um pouco mais simples nessa nova versão, para isso, agora é criado um formatador de dado com a classe **DateTimeFormatter** e a própria classe **LocalDate** tem um método **format** que retorna uma String com a data formatada no padrão passado como parâmetro.

```
public class MainLocalDateFormat {
    public static void main(String[] args) {
        LocalDate hoje = LocalDate.now();
        DateTimeFormatter formatadorBarra = DateTimeFormatter.ofPattern("dd/MM/yyyy");
        DateTimeFormatter formatadorTraco = DateTimeFormatter.ofPattern("dd-MM-yyyy");

        System.out.println("Data com /: " + hoje.format(formatadorBarra));
        System.out.println("Data com -: " + hoje.format(formatadorTraco));
    }
}
```

EXERCÍCIO

1. Após isso apresente a sua data de nascimento e dia de hoje no formato “dd/mm/yyyy”.
2. Apresente se o seu ano de nascimento foi um ano bissexto.
3. Apresente também quantos segundo se passaram desde seu nascimento.

```
public class MainLocalDateTimeTempoNasc {
    public static void main(String[] args) {
        LocalDateTime dataHoraNasc = LocalDateTime.of(1990, Month.OCTOBER, 10, 15, 25, 00);
        LocalDateTime hojeAgora = LocalDateTime.now();
        DateTimeFormatter formatadorBarra = DateTimeFormatter.ofPattern("dd/MM/yyyy");

        System.out.println("Data Nascimento: "+dataHoraNasc.format(formatadorBarra));
        System.out.println("Hoje: "+hojeAgora.format(formatadorBarra));

        System.out.println("Ano de nascimento era bissexto: "+ dataHoraNasc.toLocalDate().isLeapYear());

        Duration duracao = Duration.between(dataHoraNasc, hojeAgora);
        System.out.println("Desde o nacimento se passaram "+duracao.getSeconds()+" segundos");
    }
}
```