FCT/Unesp – Presidente Prudente
Programação Orientada a Objetos
Prof. Dr. Danilo Medeiros Eler

Trabalho Prático 02

Individual ou em grupo de no máximo três pessoas

Instruções de Envio: enviar somente para **daniloelerunesp@gmail.com** (por favor, enviar **somente** para esse e-mail). No assunto do email você deve colocar: **[POO2019] Trabalho Prático 02** No corpo do email você deve identificar o nome dos integrantes do grupo e RA. Anexar o código fonte e dados.

DICA: não envie com a extensão ZIP ou RAR para o gmail não barrar o seu email, caso haja um executável (EXE ou JAR). Você deve renomear o arquivo *zip* para *renomearParaZIP*. **Por exemplo:** trabalho02.renomearParaZIP. Alternativamente, o grupo pode enviar um link para download do trabalho.

O programa base para o trabalho pode ser encontrado em
https://github.com/daniloeler/basePOOTrabalho2

Caso haja evidencia de **cópia**, os trabalhos envolvidos terão **nota zero (0.0)**.

Data máxima para envio: O trabalho deve ser enviado por email até o dia **28/06/2019 (sexta-feira)**. Não poderei adiar a data de envio.

Especificações do trabalho

Implemente um sistema de controle de empréstimo de livros de uma biblioteca. O sistema deve cadastrar livros e usuários (Aluno e Professor), que possuem dados diferentes e também o número de dias que podem ficar com os livros emprestados. O sistema registra um empréstimo para um usuário da biblioteca, nesse momento a data de devolução prevista deve ser calculada. O empréstimo pode conter vários livros, que podem ser devolvidos em datas distintas.

Os relatórios obrigatórios são:

- **Todos Usuários:** lista todos os usuários da biblioteca, dizendo se tem livro para devolver e se tem livro com atraso;
- **Todos Alunos:** lista todos os alunos da biblioteca, dizendo se tem livro para devolver e se tem livro com atraso;
- **Todos Professores:** lista todos os professores da biblioteca, dizendo se tem livro para devolver e se tem livro com atraso;

- **Todos Livros Já Emprestados para um Usuário:** lista todos os livros já emprestados para um determinado usuário (deve-ser digitar seu código);
- **Livros não Devolvidos por um Usuário:** lista todos que ainda estão com um determinado usuário (deve-se digitar seu código), se houver atraso, o sistema deve avisar;
- **Todos os Livros:** lista todos os livros da biblioteca, relatando se está disponível ou emprestado. Também deve ser avisado se o livro estiver com atraso;
- **Livros Disponíveis:** todos os livros disponíveis (sem empréstimo) na biblioteca;
- **Livros Emprestados:** todos os livros emprestados para usuários;
- **Livros com Atraso:** todos os livros que estão com atraso;
- **Usuários com Atraso:** lista de todos os usuários que estão com livro(s) atrasado(s).

O sistema deve salvar os dados em arquivo e também permitir que eles sejam carregados (recuperados). Essa funcionalidade já está implementada para Livros. Você deve desenvolver para Usuários e para Empréstimos.

Note que há uma classe responsável pela configuração do sistema, a qual armazena o número de dias que o aluno e o professor podem ficar com os livros, bem como o caminho dos arquivos utilizados para armazenar os dados. Essa classe e a interface já estão implementadas, mas, se necessário, você pode adicionar novas configurações.

Modelo Conceitual

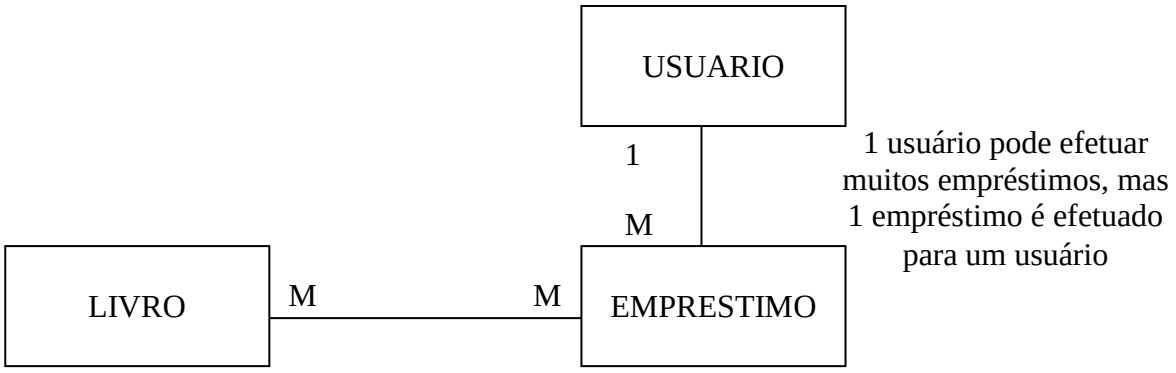
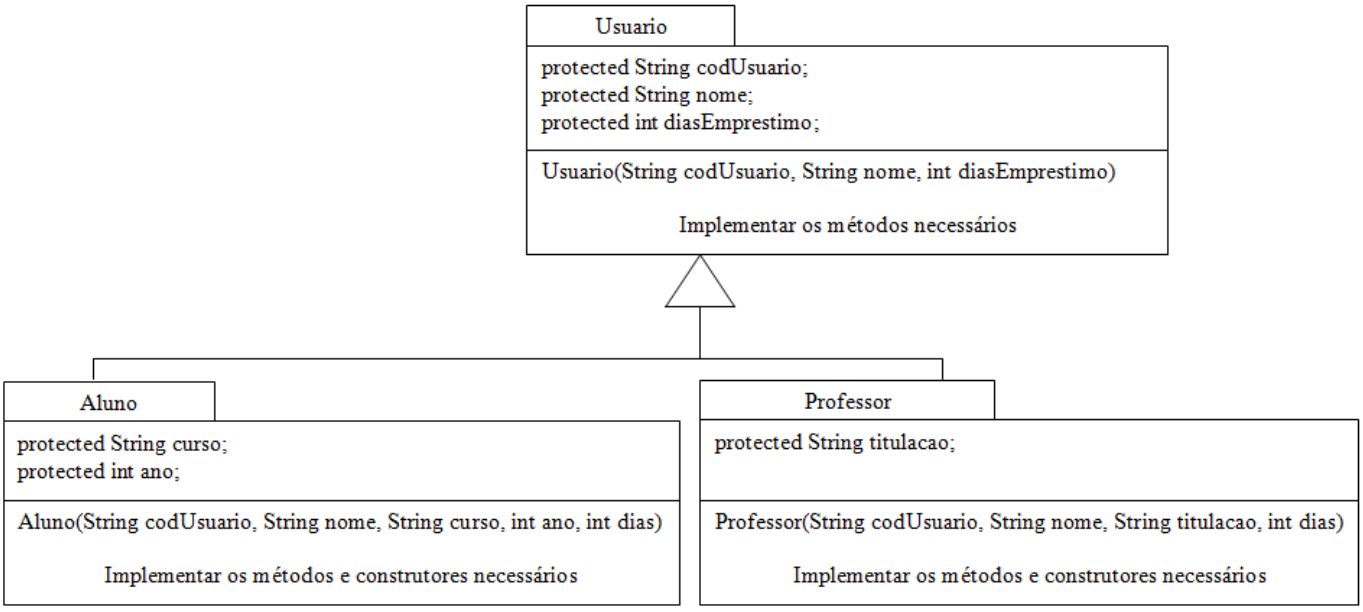
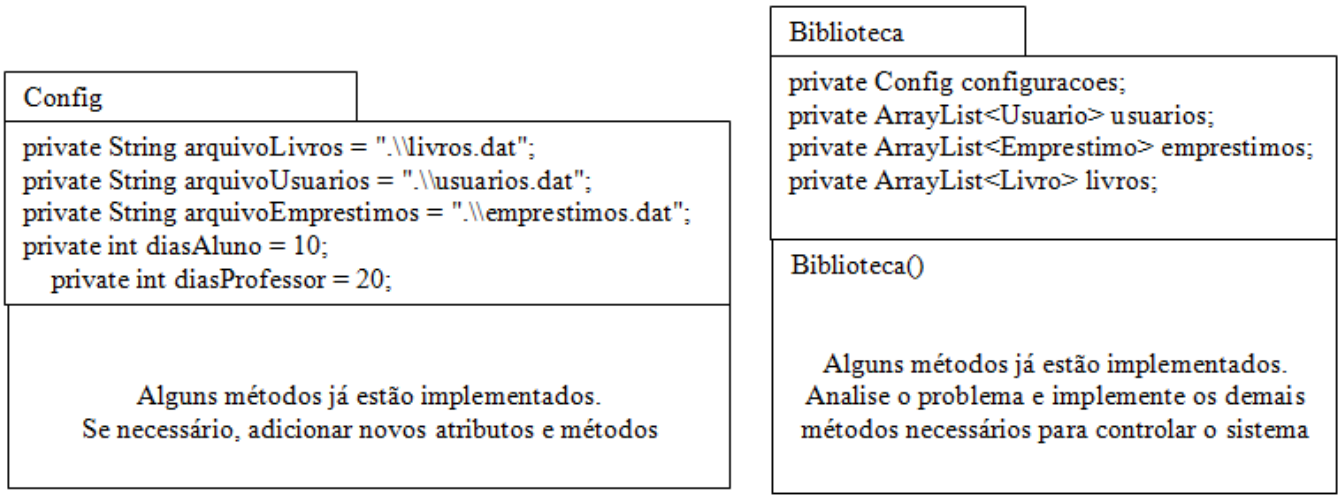
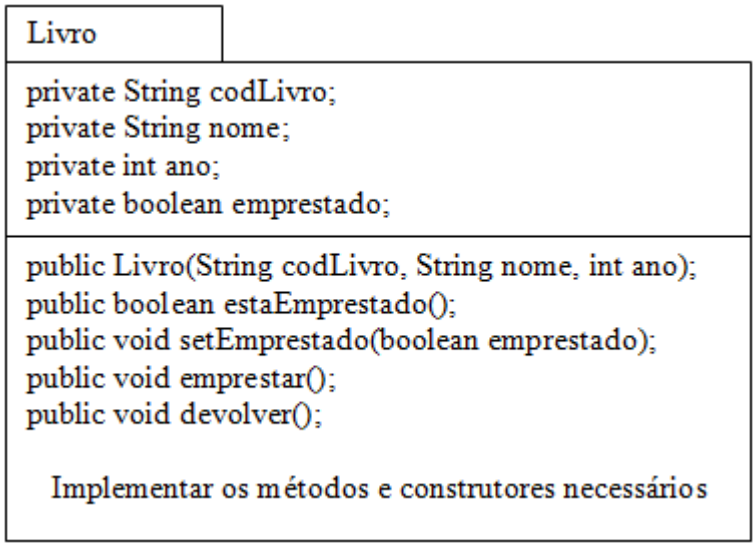


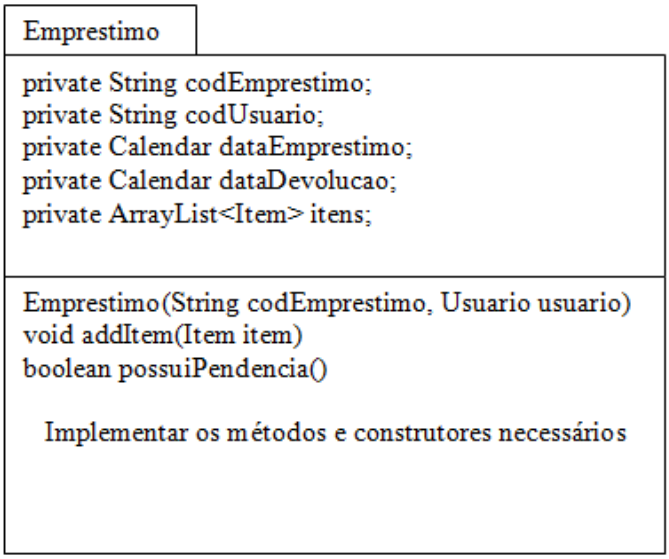
Diagrama de Classes

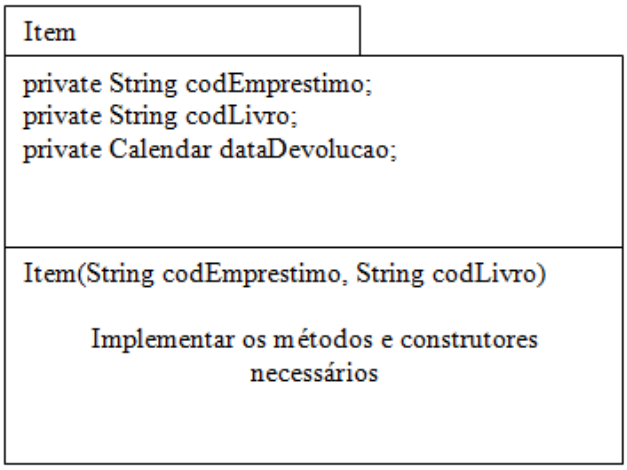
Os diagramas apresentados a seguir visam identificar o relacionamento entre as classes. Além disso, eles descrevem os atributos e alguns métodos obrigatórios. Note que alguns métodos construtores, *getters* e *setters* foram omitidos. Portanto, você deve adicionar os atributos, os construtores e os métodos conforme a sua necessidade para desenvolver o sistema. Faça também uma análise do problema e implemente os métodos necessários para o sistema.











OBSERVAÇÕES

- O sistema é uma versão simplificada de um Sistema de Biblioteca. Portanto, muitos elementos foram omitidos para facilitar o desenvolvimento;
- O sistema base que será enviado junto com esse documento é uma versão inicial do seu sistema. Muitos métodos devem ser implementados para gerenciar os dados do Sistema de Biblioteca, bem como as interfaces de usuário;
- Estou utilizando ArrayList para facilitar o gerenciamento dos dados, evitando controlar vários contadores. A maneira mais fácil seria utilizar uma tabela Hash, mas mantive o ArrayList para vocês realizarem buscas. Quem quiser utilizar array primitivo pode usar sem problemas;
- Utilizem o polimorfismo nas implementações! Isso será verificado e descontado na nota.

Exemplo de comparação de datas

```
//////Utilizando datas --- adicionando dias e comparando datas
        Emprestimo e = new Emprestimo("11", new Aluno("33", "33", "333", 1, 10));
       //  Emprestimo e = new Emprestimo("11" ,new Professor("111", "1111", "Mestre",20));
         System.out.println(e.getDataEmprestimo().getTime());
         System.out.println(e.getDataDevolucao().getTime());
        ///Comparação entre datas → método compareTo()
        //É invocado de um objeto Calendar e recebe como parâmetro outro objeto Calendar
       ///se o retorno for 0 ===> as datas são iguais
       ///se o retorno for 1 ====> quem invocou tem data superior àquela passada como parâmetro
       ///se o retorno for -1 ====> quem invocou tem data inferior àquela passada como parâmetro
         System.out.println(e.getDataEmprestimo().compareTo(e.getDataEmprestimo()));
         System.out.println(e.getDataDevolucao().compareTo(e.getDataDevolucao()));
         System.out.println(e.getDataEmprestimo().compareTo(e.getDataDevolucao()));
         System.out.println(e.getDataDevolucao().compareTo(e.getDataEmprestimo()));
```

Outras maneiras para manipular data e hora

```
System.out.println("NOVA API");
        ///nova API para trabalhar com data e hora --- pacote java.time
        LocalDate dataN = LocalDate.now();
        System.out.println(dataN);
        LocalTime t = LocalTime.now();
        System.out.println(t);
        //criar uma data
        dataN = LocalDate.of(2019, 4, 10);
        System.out.println(dataN);
        dataN = LocalDate.parse("05/04/2019",
DateTimeFormatter.ofPattern("dd/MM/yyyy"));
        System.out.println(dataN);
        t = LocalTime.of(12, 20, 0);
        System.out.println(t);
        t = LocalTime.parse("16:00", DateTimeFormatter.ofPattern("HH:mm"));
        System.out.println(t);
        //operaç?es
        LocalDate d = LocalDate.now();
        LocalDate d5 = d.plusDays(7);
        LocalDate dSemPassada = d.minus(1, ChronoUnit.WEEKS);
        System.out.println(d);
        System.out.println(d5);
        System.out.println(dSemPassada);

        LocalTime h = LocalTime.now();
        LocalTime h130 = h.plusHours(1).plusMinutes(30);
        LocalTime h2 = h.plus(5, ChronoUnit.HOURS);
        System.out.println(h);
        System.out.println(h130);
        System.out.println(h2);

        LocalDateTime dh = LocalDateTime.now();
        System.out.println(dh);
```

```
        LocalDateTime dN = dh.plusYears(1).plusDays(7).plusHours(2);
        System.out.println(dN);

        ////pegar elementos da data
        LocalDate hoje = LocalDate.now();
        System.out.println(hoje.getDayOfMonth() + " / " + hoje.getMonthValue() + " / " +
hoje.getYear());
        System.out.println(hoje.getDayOfWeek() + " / " + hoje.getMonth() + " / " +
hoje.getYear());

        ///trabalhar com período
        Period p = Period.of(1, 2, 7);
        System.out.println(p);
        LocalDate data1 = LocalDate.now();
        LocalDate data12 = data1.plus(p);
        System.out.println(data1);
        System.out.println(data12);

        Duration dd = Duration.ofMinutes(15);
        LocalTime t1 = LocalTime.now();
        LocalTime t2 = t1.minus(dd);
        System.out.println(t1);
        System.out.println(t2);

        hoje = LocalDate.now();
        LocalDate aniv = LocalDate.parse("1980-06-28");

        Period pp = Period.between(aniv, hoje);
        int years = pp.getYears();
        int months = pp.getMonths();
        int days = pp.getDays();
        System.out.println(years + " " + months + " " + days);

        LocalTime t11 = LocalTime.now();
        LocalTime t22 = LocalTime.parse("12:00:00");

        Duration ddd = Duration.between(t22, t11);
        long seconds = ddd.getSeconds();
        System.out.println(seconds);

        ///intervalos
        System.out.println(ChronoUnit.MONTHS.between(aniv, hoje));
        System.out.println(ChronoUnit.YEARS.between(aniv, hoje));
        t22 = LocalTime.now();
        System.out.println(t11);
        System.out.println(t22);
        System.out.println(ChronoUnit.SECONDS.between(t22, t11));

        Instant begin = Instant.now();
        Instant end = Instant.now();
        Duration tempo = Duration.between(begin, end);
        long seg = tempo.getSeconds();
```

```
System.out.println(seg);
System.out.println(tempo.getNano());
```