

Criar um projeto -> Aula13.

Criar no pacote entity as classes Jogador, Equipe, Pontuacao e GerenteProjeto. Na classe Jogador criar os atributos(idJogador, nome e pontos). Fazer o relacionamento do jogador com a classe Pontuacao e atributo pontuacao. Criar construtor vazio, o construtor cheio, o toString sem a impressão da senha e os getters e setters.

```
package entity;
```

```
public class Jogador {
```

```
    private Integer idJogador;
```

```
    private String nome;
```

```
    private transient Integer pontos = 0;
```

```
    private Pontuacao pontuacao;
```

```
    public Jogador() {
```

```
    }
```

```
    public Jogador(Integer idJogador, String nome, Integer  
pontos) {
```

```
        this.idJogador = idJogador;
```

```
        this.nome = nome;
```

```
        this.pontos = pontos;
```

```
    }
```

```
@Override
```

```
public String toString() {
```

```
    return "Jogador [idJogador=" + idJogador + ", nome=" +  
nome + ", pontos=" + pontos + "];"
```

```
}
```

```
public Integer getIdJogador() {
```

```
    return idJogador;
```

```
}
```

```
public void setIdJogador(Integer idJogador) {
```

```
    this.idJogador = idJogador;
```

```
}
```

```
public String getNome() {
```

```
    return nome;
```

```
}
```

```
public void setNome(String nome) {
```

```
    this.nome = nome;
```

```
}
```

```
public Integer getPontos() {
    return pontos;
}
public void setPontos(Integer pontos) {
    this.pontos = pontos;
}
public Pontuacao getPontuacao() {
    return pontuacao;
}
public void setPontuacao(Pontuacao pontuacao) {
    this.pontuacao = pontuacao;
}
}
```

Na classe Pontuacao, criar os atributos idCartao e cor. Relacionar a classe Pontuação com a classe Jogador e seu atributo jogador. Criar os construtores vazio e cheio, toString e getters e setters.

```
package entity;

public class Pontuacao {

    private Integer idCartao;
    private String cor;

    private Jogador jogador;

    public Pontuacao() {

    }

    public Pontuacao(Integer idCartao, String cor) {
        super();
        this.idCartao = idCartao;
        this.cor = cor;
    }

    @Override
    public String toString() {
        return "Pontuacao [idCartao=" + idCartao + ", cor=" +
cor + " ]";
    }

    public Integer getIdCartao() {
```

```
        return idCartao;
    }
    public void setIdCartao(Integer idCartao) {
        this.idCartao = idCartao;
    }
    public String getCor() {
        return cor;
    }
    public void setCor(String cor) {
        this.cor = cor;
    }

    public Jogador getJogador() {
        return jogador;
    }

    public void setJogador(Jogador jogador) {
        this.jogador = jogador;
    }
}
```

Criar a classe GerenteProjeto com os atributos(código e nome). Relacionar a classe GerenteProjeto com a lista de Equipe. Significando que a classe GerenteProjeto tem uma lista de Equipe. Criar os construtores vazio e cheio, toString e getters e setters.

```
package entity;

import java.util.ArrayList;
import java.util.List;

public class GerenteProjeto {

    private Integer codigo;
    private String nome;

    private List<Equipe> equipes;

    public GerenteProjeto() {

    }
}
```

```
public GerenteProjeto(Integer codigo, String nome,
List<Equipe> equipes) {
    this.codigo = codigo;
    this.nome = nome;
    this.equipes = equipes;
}

@Override
public String toString() {
    return "GerenteProjeto [codigo=" + codigo + ", nome=" +
nome + ", equipes=" + equipes + "];"
}

public Integer getCodigo() {
    return codigo;
}
public void setCodigo(Integer codigo) {
    this.codigo = codigo;
}
public String getNome() {
    return nome;
}
public void setNome(String nome) {
    this.nome = nome;
}
public List<Equipe> getEquipes() {
    return equipes;
}

public void add(Equipe e){
    if (e==null){
        equipes = new ArrayList<Equipe>();
    }
    equipes.add(e);
}
}
```

Criar a classe Equipe com os atributos (idEquipe e nomeFuncionario). Relacionar a classe Equipe com a classe GerenteProjeto e o atributo gerente. Criar os construtores vazio, cheio com todos os atributos e cheio sem o atributo de gerente. Criar o toString e os getters e setters.

```
package entity;

public class Equipe {

    private Integer idEquipe;
    private String nomeFuncionario;

    private GerenteProjeto gerente;

    public Equipe() {

    }

    public Equipe(Integer idEquipe, String nomeFuncionario,
                  GerenteProjeto gerente) {
        super();
        this.idEquipe = idEquipe;
        this.nomeFuncionario = nomeFuncionario;
        this.gerente = gerente;
    }

    public Equipe(Integer idEquipe, String nomeFuncionario) {
        super();
        this.idEquipe = idEquipe;
        this.nomeFuncionario = nomeFuncionario;
    }

    @Override
    public String toString() {
        return "Equipe [idEquipe=" + idEquipe + ",
nomeFuncionario=" + nomeFuncionario + ", gerente=" + gerente +
        "];"
    }

    public Equipe(GerenteProjeto gerente) {
        this.gerente = gerente;
    }

    public Integer getIdEquipe() {
        return idEquipe;
    }

    public void setIdEquipe(Integer idEquipe) {
        this.idEquipe = idEquipe;
    }
}
```

```
}  
public String getNomeFuncionario() {  
    return nomeFuncionario;  
}  
public void setNomeFuncionario(String nomeFuncionario) {  
    this.nomeFuncionario = nomeFuncionario;  
}  
public void setGerente(GerenteProjeto gerente) {  
    this.gerente = gerente;  
}
```

No método `getGerente`, colocamos a condição de que se o gerente for nulo, o atributo dá um espaço de memória pra o gerente ser criado e retorna esse gerente.

```
public GerenteProjeto getGerente() {  
    if (gerente == null) {  
        gerente = new GerenteProjeto();  
    }  
    return gerente;  
}
```

No método de teste, chamamos a classe `GerenteProjeto`, criamos o objeto e damos espaço de memória. Em seguida setamos, ou seja, passamos os dados de código e um nome. Chamamos a classe `Equipe` e o objeto, setamos os dados de equipe(código, nome e gerente, se houver). Na impressão, pediremos para imprimir o nome da equipe e o nome do gerente se houver.

```
public static void main(String[] args) {  
  
    GerenteProjeto gp = new GerenteProjeto();  
    gp.setCodigo(1000);  
    gp.setNome("stuart");  
  
    Equipe e1 = new Equipe(10, "lu", gp);  
    Equipe e2 = new Equipe(12, "anthony");  
    Equipe e3 = new Equipe(14, "garrito", gp);  
  
    System.out.println(e1.getNomeFuncionario() + ", GP=" +  
e1.getGerente().getNome());  
    System.out.println(e2.getNomeFuncionario() + ", GP=" +  
e2.getGerente().getNome());  
}
```

```
        System.out.println(e3.getNomeFuncionario() + ", GP=" +
e3.getGerente().getNome());
    }
}
```

No console...

```
lu, GP=stuart
anthony, GP=null
garrito, GP=stuart
```

Criar um pacote control e nele criar uma classe CalcularPontuacao para os cálculos dos pontos dos jogadores.

```
package control;

import entity.Jogador;
import entity.Pontuacao;

public class CalcularPontuacao {
```

Criar um método adicionar, passando como parâmetro a classe Jogador, a Pontuacao e uma variável do tipo inteiro chamado pontos. Relacionar o objeto da classe Jogador, passando o método setPontuacao e parâmetro p. fazer o mesmo para o inverso, relacionar o objeto p de Pontuacao com j do Jogador. Sendo um relacionamento bidirecional, as duas classes se vêem. Chamar o objeto j e setar os Pontos do jogador onde irá somar os pontos já existentes com os pontos adicionados.

```
    public void adicionar(Jogador j, Pontuacao p, Integer
pontos) {
        j.setPontuacao(p);
        p.setJogador(j);
        j.setPontos( (j.getPontos() + pontos) );
    }
}
```

No teste main, criar os jogadores com seus dados, criar as pontuações também com seus dados.

```
public static void main(String[] args) {  
    Jogador j1 = new Jogador(10, "leandro", 0);  
    Jogador j2 = new Jogador(11, "noe", 0);  
  
    Pontuacao p1 = new Pontuacao(1001, "azul");  
    Pontuacao p2 = new Pontuacao(1002, "vermelho");
```

Chamar a classe de calculo, `CalcularPontuacao`, criar objeto e dar espaço de memória. Passar o objeto com o método `adicionar` e os parâmetros de relação: Jogador `j1` com cartela de pontuação `p1` e quantidade de pontos. Na impressão, na primeira linha imprimir o nome do jogador e a cartela, na segunda, a quantidade final de pontos.

```
    CalcularPontuacao cp = new CalcularPontuacao();  
    cp.adicionar(j1, p1, 30);  
    cp.adicionar(j1, p1, 100);  
    cp.adicionar(j2, p2, 50);  
  
    System.out.println(j1.getNome() + ", " +  
j1.getPontuacao().getIdCartao());  
    System.out.println("Pontos :" + j1.getPontos());  
    System.out.println(j2.getNome() + ", " +  
j2.getPontuacao().getIdCartao());  
    System.out.println("Pontos :" + j2.getPontos());  
  
    }  
}
```

No console...

```
leandro,1001  
Pontos :130  
noe,1002  
Pontos :50
```