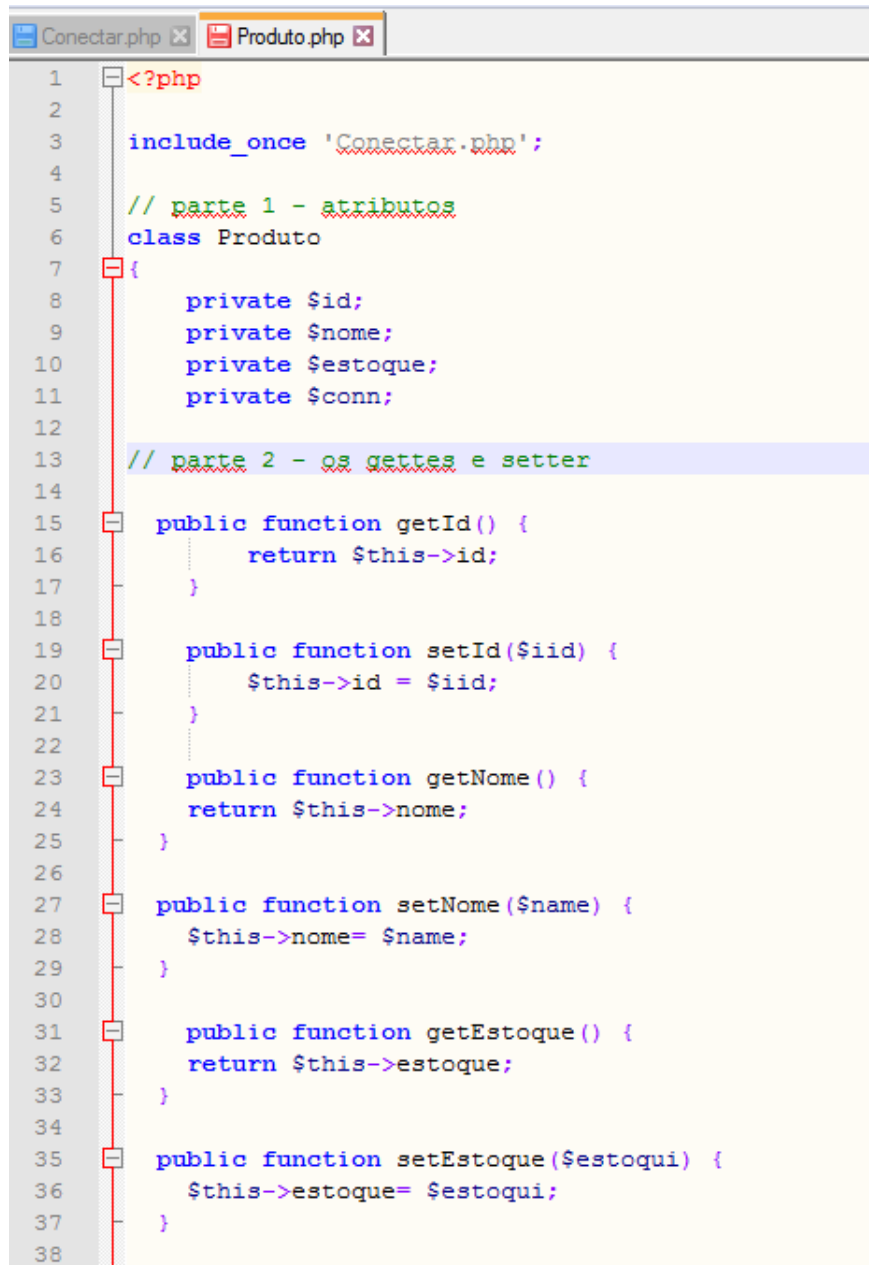


### === Classe de Modelagem Produto

Vamos criar, agora, a classe que será a base para o banco de dados executar as manutenções. Esta classe será dividida em partes: atributos, getters e setters e por último os métodos.

Crie um arquivo **PHP**, com o nome **Produto.php** dentro da pasta **acessoBD**. Comece a editar a classe Produto, iniciando com os atributos e os getters e setters:



```
1 <?php
2
3 include_once 'Conectar.php';
4
5 // parte 1 - atributos
6 class Produto
7 {
8     private $id;
9     private $nome;
10    private $estoque;
11    private $conn;
12
13    // parte 2 - os gettes e setter
14
15    public function getId() {
16        return $this->id;
17    }
18
19    public function setId($iid) {
20        $this->id = $iid;
21    }
22
23    public function getNome() {
24        return $this->nome;
25    }
26
27    public function setNome($name) {
28        $this->nome= $name;
29    }
30
31    public function getEstoque() {
32        return $this->estoque;
33    }
34
35    public function setEstoque($estoqui) {
36        $this->estoque= $estoqui;
37    }
38 }
```

Agora na sequência, os métodos referentes para cada ação executada pelo menu:

Obs.: Perceba que utilizamos o **try** e **catch**, no catch é tratada a exceção PDO, esta retornará qualquer exceção relacionada ao acesso PHP e MySQL. Este é mais um dos motivos para utilizar a extensão PDO, afinal todo o gerenciamento de acesso e controle ao MySQL é trabalhada por ela.

```

39 //parte 3 - métodos
40
41 function salvar()
42 {
43     try
44     {
45         $this->conn = new Conectar();
46         $sql = $this->conn->prepare("insert into produto values (null,?,?)");
47         @$sql->bindParam(1, $this->getNome(), PDO::PARAM_STR);
48         @$sql->bindParam(2, $this->getEstoque(), PDO::PARAM_STR);
49         if($sql->execute() == 1)
50         {
51             return "Registro salvo com sucesso!";
52         }
53         $this->conn = null;
54     }
55     catch(PDOException $exc)
56     {
57         echo "Erro ao salvar o registro. " . $exc->getMessage();
58     }
59 }

```

O método “salvar” fará a inclusão de novos dados digitados na tabela, será utilizado na rotina “cadastrar produto”.

```

60
61 function alterar()
62 {
63     try
64     {
65         $this->conn = new Conectar();
66         $sql = $this->conn->prepare("select * from produto where id = ?"); // informei o ? (parametro)
67         @$sql->bindParam(1, $this->getId(), PDO::PARAM_STR); // inclui esta linha para definir o parametro
68         $sql->execute();
69         return $sql->fetchAll();
70         $this->conn = null;
71     }
72     catch(PDOException $exc)
73     {
74         echo "Erro ao alterar. " . $exc->getMessage();
75     }
76 }
77
78 function alterar2()
79 {
80     try
81     {
82         $this->conn = new Conectar();
83         $sql = $this->conn->prepare("update produto set nome = ?, estoque = ? where id = ?");
84         @$sql->bindParam(1, $this->getNome(), PDO::PARAM_STR);
85         @$sql->bindParam(2, $this->getEstoque(), PDO::PARAM_STR);
86         @$sql->bindParam(3, $this->getId(), PDO::PARAM_STR);
87         if($sql->execute() == 1)
88         {
89             return "Registro alterado com sucesso!";
90         }
91         $this->conn = null;
92     }
93     catch(PDOException $exc)
94     {
95         echo "Erro ao salvar o registro. " . $exc->getMessage();
96     }
97 }
98

```

A rotina de alteração possui uma particularidade: os dados que estão no banco primeiro serão mostrados em caixa de texto (método alterar) e após a digitação das alterações, estas serão gravadas na tabela (método alterar2).

```
98
99 function consultar()
100 {
101     try
102     {
103         $this->conn = new Conectar();
104         $sql = $this->conn->prepare("select * from produto where nome like ?"); // informei o ?
105         @$sql->bindParam(1, $this->getNome(), PDO::PARAM_STR); // inclui esta linha para definir o parametro
106         // @$sql->bindParam(1, $this->getNome()."%", PDO::PARAM_STR);
107         $sql->execute();
108         return $sql->fetchAll();
109         $this->conn = null;
110     }
111     catch(PDOException $exc)
112     {
113         echo "Erro ao executar consulta. " . $exc->getMessage();
114     }
115 }
116
```

O método consultar irá buscar na tabela os dados referente ao parâmetro pesquisado, neste caso o campo nome. Perceba que o método retorna o fetchAll (linha 108), este método PDO tem por finalidade retornar uma matriz carregada com todos os registros da table, conforme o comando SELECT configurado anteriormente (linha 104).

```
116
117 function exclusao()
118 {
119     try
120     {
121         $this->conn = new Conectar();
122         $sql = $this->conn->prepare("delete from produto where id = ?"); // informei o ? (parametro)
123         @$sql->bindParam(1, $this->getId(), PDO::PARAM_STR); // inclui esta linha para definir o parametro
124         if($sql->execute() == 1)
125         {
126             return "Excluido com sucesso!";
127         }
128         else
129         {
130             return "Erro na exclusao !";
131         }
132         $this->conn = null;
133     }
134     catch(PDOException $exc)
135     {
136         echo "Erro ao excluir. " . $exc->getMessage();
137     }
138 }
139
140
141
```

Parecido com o anterior, o método exclusão irá excluir (deletar) da tabela o registro referente ao parâmetro pesquisado, neste caso o campo id.

```
141
142 function listar()
143 {
144     try
145     {
146         $this->conn = new Conectar();
147         $sql = $this->conn->query("select * from produto order by nome");
148         $sql->execute();
149         return $sql->fetchAll();
150         $this->conn = null;
151     }
152     catch(PDOException $exc)
153     {
154         echo "Erro ao executar consulta. " . $exc->getMessage();
155     }
156 }
157
158 } // encerramento da classe Produto
```

Já este método, como o próprio nome deduz, será responsável por apresentar na tela uma lista apresentando todos os registros existentes nesta tabela (linha 147).

Perceba que na linha 158 temos a chave “}” fechando a classe que foi aberta na linha 7.

Nas próximas aulas aprenderemos a criar as “telas” que irão chamar os métodos criados nesta classe de modelagem.