

Exercise 1:

Start with the following list of numbers:

```
const numbers = [1, 2, 4, 7];
```

Print the square of each number: 1, 4, 16 and 49.

1. First, use a for loop.
2. Solve the problem again using a for-in loop.

Exercise 2:

1. Create a class named User as follows:

```
class User {  
  int id = 213;  
  String name = 'Fatima';  
}
```

2. Create an object using this class
3. Print the object you created.
4. What will appear in the screen.
5. Define a toString method.
6. Print the object again
7. What will appear in the screen.

Exercise 3:

1. Create a constructor using the long-form, Short-Form, name constructor, and forwarding methods to initialize the id and the name of the user
2. Use the constructors to create an object from the class
8. Define a toString method.
9. Print the object again for each constructor you used

```
1 void main() {  
2   User user1=User(123,'Hala');  
3   print(user1);  
4   User user2=User.shortFone(456 , 'Hala');  
5   print(user2);  
6   User user3=User.nameConstructor(789, 'Hala');  
7   print(user3);  
8   User user4=User.forwardingconstructor('Hala');  
9   print(user4);  
10 }  
11  
12 class User {  
13   int id = 65;  
14   String name= 'x';  
15   User(int id ,String name){  
16     this.id=id;  
17     this.name=name;  
18   }  
19  
20   User.shortFone(this.id,this.name);  
21   User.nameConstructor(int id,String name){  
22     this.id=id;  
23     this.name=name;  
24   }  
25   User.forwardingconstructor(String name):this(0,name);  
26   toString(){  
27     return 'User(id$id,name:$name)';  
28   }  
29 }  
30
```

Console output:

```
User(id123, name:Hala)  
User(id456, name:Hala)  
User(id789, name:Hala)  
User(id0, name:Hala)
```

Exercise 4:

1. Define one of the previous constructor with name parameters.
2. Create object and print it.

```
1 void main() {  
2   User user1=User(123,'Hala');  
3   print(user1);  
4   User user2=User.shortFone(456 , 'Hala');  
5   print(user2);  
6   User user3=User.nameConstructor(id:789,name: 'Hala');  
7   print(user3);  
8   User user4=User.forwardingconstructor('Hala');  
9   print(user4);  
10 }  
11  
12 class User {  
13   int id = 65;  
14   String name= 'x';  
15   User(int id ,String name){  
16     this.id=id;  
17     this.name=name;  
18   }  
19  
20   User.shortFone(this.id,this.name);  
21   User.nameConstructor({int id=00,String name='sara'}){  
22     this.id=id;  
23     this.name=name;  
24   }  
25   User.forwardingconstructor(String name):this(0,name);  
26   toString(){  
27     return 'User(id$id,name:$name)';  
28   }  
29 }  
30
```

Console output:

```
User(id123, name:Hala)  
User(id456, name:Hala)  
User(id789, name:Hala)  
User(id0, name:Hala)
```

Exercise 5:

1. Define the previous class with private instance variables
2. Use set and get methods

dartpad - Search

DartPad

Search - سطور فلات بورد

مراجعة سجل التغيير 4

Search - جميل

(no subject) - 442050639@std.p...

Inbox - 442050639@std.p...

https://dartpad.dev/?

DartPad

<> New Pad

Reset

Format

Install SDK

obsidian-halo-3273

total edit

Samples

1* void main() {
2 User user=User(123,'Hala');
3 print(user);
4 user._id=99;
5 user._name='wafa';
6 print(user);
7
8 }
9
10 class User {
11 int _id = 65;
12 String _name='k';
13 User(int id ,String name){
14 this._id=id;
15 this._name=name;
16 }
17 String get name1 => _name;
18 set name1 (String x){
19 _name=x;
20 }
21 int get id1 => _id;
22 set id1 (int y){
23 _id=y;
24 }
25 String toString(){
26 return 'UserwithPrivate(id\$_id,name:\$name)';
27 }
28 }
29
30

Run

Console

UserwithPrivate(id123,name:Hala)
UserwithPrivate(id99,name:wafa)

Documentation

class User

Info

line 13 • Unnecessary 'this.' qualifier. (view docs)
Try removing 'this.'

Info

line 14 • Unnecessary 'this.' qualifier. (view docs)
Try removing 'this.'

Info

line 16 • Unnecessary use of getter and setter to wrap a field. (view docs)
Try removing the getter and setter and renaming the field.

Info

line 20 • Unnecessary use of getter and setter to wrap a field. (view docs)
Try removing the getter and setter and renaming the field.

Info

line 24 • The member 'toString' overrides an inherited member but isn't annotated with '@override'. (view docs)
Try adding the '@override' annotation.

Privacy notice

Send feedback

stable channel

5 issues

hide

Based on Flutter 3.13.4 Dart SDK 3.1.2

ENG

2:37 PM

10/10/2023