# Problem 2. Press House

### Your Task

Your need to create several classes for **Press House**. Implement the following classes:

**Article, ShortReports, BookReview.**

### Article

**constructor(title, content)**

Should have these **2** properties:

* **title – string**
* **content – string**

**toString()**

This **function** should **return** the **title** and the **content:**

**"** **Title: {title}**

**Content: {content}"**

### ShortReports

Class **ShortReports** inherits class **Article**.

**constructor(title, content,** **originalResearch)**

Should have these **4** properties:

* **title – string,**
* **content – string,**
* **originalResearches – object** with properties **title** and **author**
* **comments – array of strings**

As we create a short reports here we have a length limit for the **content** property – it should be less than 150 symbols, otherwise throw an error with the next message:

**"Short reports content should be less then 150 symbols."**

The property should have the both required properties , otherwise **throw error** with this message:

**"The original research should have author and title."**

**addComment(comment)**

This **function** should receive single **comment** like **string**, add it to the **comments** array and **return** a message:

**"The comment is added."**

**toString()**

This **function** should extend the **toString** method of class **Article** adding same more lines like**:**

**"Original Research: { title } by { author }"**

And if there are any comments you should print on a new line

**"Comments:"**

and then all comments each on a new line.

Note: For more information see the examples below!

### BookRewiew

Class **BookReview** inherits class **Article**.

**constructor(title, content,** **book)**

Should have these **4** properties:

* **title – string,**
* **content – string,**
* **book – object** with properties **name** and **author,**
* **clients – array of objects,**

The client object should have the following structure **{clientName, orderDescription}**.

**addClient(clientName,** **orderDescription)**

This **function** should receive **clientName** and **orderDescription** as strings. Here you should check our clients array and if we already have this order from the same client throw error with next message:

**"This client has already ordered this review."**

Otherwise we add our client object into the **clients** array and return a message:

**"{ clientName } has ordered a review for { book name }"**

**toString()**

This **function** should extend the **toString()** method of class **Article** adding same more lines like**:**

**"Book: { book name }"**

And if there are any orders you should print all orders each on a new line:

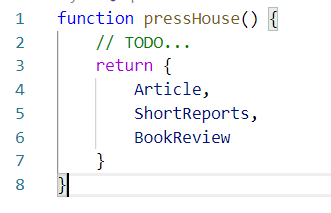
**"Orders:**

**{ clientName } - { orderDescription }"**

Note: For more information see the examples below!

### Submission

Submit your **pressHouse** function**.**



### Examples

This is an example how the code is **intended to be used**:

|  |
| --- |
| Sample code usage |
| let classes = solveClasses();  let lorem = new classes.Article("Lorem", "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce non tortor finibus, facilisis mauris vel…");  console.log(lorem.toString());  ------------------------------  let short = new classes.ShortReports("SpaceX and Javascript", "Yes, its damn true.SpaceX in its recent launch Dragon 2 Flight has used a technology based on Chromium and Javascript. What are your views on this ?", { title: "Dragon 2", author: "wikipedia.org" });  console.log(short.addComment("Thank god they didn't use java."))  short.addComment("In the end JavaScript"s features are executed in C++ — the underlying language.")  console.log(short.toString());  ------------------------------  let book = new classes.BookReview("The Great Gatsby is so much more than a love story", "The Great Gatsby is in many ways similar to Romeo and Juliet, yet I believe that it is so much more than just a love story. It is also a reflection on the hollowness of a life of leisure. ...", { name: "The Great Gatsby", author: "F Scott Fitzgerald" });  console.log(book.addClient("The Guardian", "100 symbols"));  console.log(book.addClient("Goodreads", "30 symbols"));  console.log(book.toString()); |
| Corresponding output |
| Title: Lorem  Content: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce non tortor finibus, facilisis mauris vel, ultricies est. Phasellus id pellentesque risus. Morbi aliquet at lectus ac malesuada. Morbi eu erat orci. Donec id turpis elit. Donec iaculis sapien odio, sit amet cursus lacus rutrum sit amet. Cras ac urna sapien. Pellentesque porta mauris ac dolor commodo, congue condimentum orci varius. Ut ultrices pretium commodo. Aenean facilisis mattis facilisis.  ----------------------  The comment is added.  Title: SpaceX and Javascript  Content: Yes, its damn true.SpaceX in its recent launch Dragon 2 Flight has used a technology based on Chromium and Javascript. What are your views on this ?  Original Research: Dragon 2 by wikipedia.org  Comments:  Thank god they didn't use java.  In the end JavaScript's features are executed in C++ — the underlying language.  ----------------------  The Guardian has ordered a review for The Great Gatsby  Goodreads has ordered a review for The Great Gatsby  Title: The Great Gatsby is so much more than a love story  Content: The Great Gatsby is in many ways similar to Romeo and Juliet, yet I believe that it is so much more than just a love story. It is also a reflection on the hollowness of a life of leisure. ...  Book: The Great Gatsby  Orders:  The Guardian - 100 symbols  Goodreads - 30 symbols |

# Problem 3. Bank

class Bank {  
 *// TODO: implement this class...*  
}

Your Task

Write a Class Bank, Which Implements the Following Functionality:

Functionality

#### constructor (bankName)

Receives 1 parameter at initialization of the class (bankName), and should be set as private property.

Should have these **2** properties:

* bankName - private property of type string
* allCustomers - initially an empty array

#### newCustomer (customer)

The customer is of type object {firstName, lastName, personalId}.

* Check if the customer is already a customer of the bank. If so you should throw an Error:

”{firstName} {lastName} is already our customer!”

* Otherwise this function should add the customer as new one and return the customer details.

#### depositMoney (personalId, amount)

Both the personalId and the amount are numbers.

* Check if the given **personalId** corresponds to a customer in the **customers** array, if not **throw a new error**:

“We have no customer with this ID!”

* Otherwise **add the amount** to the corresponding customer in a property named **totalMoney** and **store the transaction information** to this customer (for more clarity see the example below and the hints), then **return the total money** of the corresponding customer and a dollar sign:

“{totalMoney}$”

#### withdrawMoney (personalId, amount)

Both the personalId and the amount are numbers.

* Check if the given **personalId** corresponds to a customer in the **customers** array, if not **throw a new error**:

“We have no customer with this ID!”

* If there is a customer with the given personalId, check if the customer has enough money in his account, to withdraw the given amount. If the money is not enough throw a new error:

“{firstName} {lastName} does not have enough money to withdraw that amount!”

* Otherwise subtract the **amount** from the **totalMoney** of the customerand store the **transaction information** to this customer, then **return the total money** of the corresponding customer and a dollar sign:

“{totalMoney}$”

#### customerInfo (personalId)

The personalId is of type number.

* Check if the given **personalId** corresponds to a customer in the **customers** array, if not **throw a new error**:

“We have no customer with this ID!”

* Otherwise return the whole information for the customer in the following format:

**“Bank name: {bankName}**

**Customer name: {firstName} {lastName}**

**Customer ID: {personalId}**

**Total Money: {totalMoney}$**

**Transactions:**

**n. {firstName} {lastName} made deposit of {amount}$!**

**...**

1. {firstName} {lastName} withdrew {amount}$!
2. {firstName} {lastName} made deposit of {amount}$!”

**Transaction information** contains information about:

* **number** of the transaction in descending order;
* **names** (firstName, lastName);
* if the transaction is **deposit/withdraw**;
* **amount** of the transaction.

Examples

This is an example how the code is intended to be used:

|  |
| --- |
| Sample code usage |
| let bank = new Bank("SoftUni Bank");  console.log(bank.newCustomer({firstName: "Svetlin", lastName: "Nakov", personalId: 6233267}));  console.log(bank.newCustomer({firstName: "Mihaela", lastName: "Mileva", personalId: 4151596}));  bank.depositMoney(6233267, 250);  console.log(bank.depositMoney(6233267, 250));  bank.depositMoney(4151596,555);  console.log(bank.withdrawMoney(6233267, 125));  console.log(bank.customerInfo(6233267)); |
| Corresponding output |
| **{ firstName: "Svetlin", lastName: "Nakov", personalId: 6233267 }**  **{ firstName: "Mihaela", lastName: "Mileva", personalId: 4151596 }**  500$  375$  Bank name: SoftUni Bank  Customer name: Svetlin Nakov  Customer ID: 6233267  Total Money: 375$  Transactions:  3. Svetlin Nakov withdrew 125$!  2. Svetlin Nakov made depostit of 250$!  1. Svetlin Nakov made depostit of 250$! |

*GOOD LUCK!😊*