# TS Advanced: Retake Exam - 25.01.20

# Problem 3. Bank Cards

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

### Your Task

### Write a BankCards class, which supports the described functionality below. Firstly, import the ICardService interface and the BankOffice class. The BankCards class should extend an abstract class, described below and implement the ICardService interface

### Create an abstract class CardService

### Functionality

#### generateCardNumber({min}, {max})

Both the **min** and the **max** are **numbers**

Check if the min number **is below 0** and if it is **larger** than the **max number**, if so, throw a new error:

**"The numbers should start from 0 and min cannot be larger than max"**

Check if the max number **is above 9** and if it is **smaller** than the **min number**, if so, throw a new error:

**"The numbers should not be larger than 0 and max cannot be smaller than max"**

This method should **return** a random **9-digit number**. In order to generate it use the following formula:

**Math.floor(Math.random() \* (max - min + 1)) + min**

#### validatePIN({PIN})

The **PIN** is a **number**

Check if the **PIN** has **exactly** **4** digits, if so return the **PIN** code.

Otherwise throw a new error:

**"Invalid PIN"**

### Create the BankCards class

### Functionality

#### constructor()

Receives **no** parameters at initialization of the class, but has one public property:

* **cardsArray** - empty array of objects

#### get randomCard()

Generate a **random** **digit (0 - 100)**.

**Hint**: Use the formula above to do so.

#### createCard({cardType}, {PIN})

The **cardType** is a string and the **PIN** is a number

* Generate a bank card number by using the method from the abstract class.
* Check if the given PIN is valid, again by using the method from the abstract class, if not throw an error

**"Invalid PIN"**

* Otherwise push an object with the **cardType**, the **card number** and the **PIN**

#### issueCard({customerID})

The **customerID** is a number

* Check if the given **customerID** corresponds to a customer in the **customers** array (from the imported **BankOffice** class), if not throw a new error:

**"Invalid customer."**

* After that get a card from the card array (use the randomly generated card number as index). Check if there is a card on that index, if not throw a new error:

**"Invalid bank card"**

* Otherwise add the bank card to the found customer with the given **customerID** and delete it from the array

#### showCustomersWithCards()

Take all customers that have been issued bank cards and return a message in the following format:

#### "{firstName} {lastName} has {cardType} with number {cardNumber}

**{firstName} {lastName} has {cardType} with number {cardNumber}**

. . . "

### Examples

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

|  |
| --- |
| Sample code usage |
| **let card = new BankCards();**  **card.createCard(**"**Debit**"**, 1111);**  **card.createCard(**"**Visa**"**, 2222);**  **card.createCard(**"**Mastercard**"**, 3333);**  **card.createCard(**"**Maestro**"**, 4444);**  **card.issueCard(1);**  **card.issueCard(2);**  **card.issueCard(3);**  **console.log(card.showCustomersWithCards());** |
| Corresponding output |
| Ivan Ivanov has Debit with number - 444021413  Petar Petrov has Mastercard with number - 44410410  Todor Todorov has Visa with number - 110200410 |