OOP

Objektno orijentisano programiranje (OOP), pocinje od ideje “klasa”, pri cemu je klasa nacrt ili sablon iz kojeg se kreiraju objekti. Objekte kreiramo koristeci kljucnu rijec “new”.

Klasa je sustinski nacrt strukture koju ce objekti kreirani iz te klase imati. Na primjer, ako kreiramo klasu:

```
class Person {
	name = “”;
	constructor(name, email, phone) {
		this.name = name;
		this.email = email;
		this.phone = phone;
	}

	setName(name) {
		this.name = name;
	}

}
```

koristeci ovu klasu mozemo kreirati objekte, kao npr;

```
const person = new Person(“Branimir”, “branimirdragicevic1@gmail.com”, “+38766088929”);
person.name
```

ali takodje i:

```
const anotherPerson = new Person(“Somebody”, “somebody@mail.com”, “00123456789”);
```

Ono sto smo dobili sa ovim jeste zagarantovana struktura objekata (donekle) koji se kreiraju iz klase Person. I dalje nismo sprijeceni da dinamcki dodamo ili izbrisemo neke polje iz ovako kreiranih objekata, ali bar inicijalno smo sigurni da imaju odredjenu strukturu.

Druga bitna stvar - dobili smo re-upotrebljivu strukturu koju mozemo koristiti za generisanje proizvoljnog broja objekata (odnosno onoliko koliko nam memorija masine dozvoljava).

Pored polja, klase mogu sadrzati i metode. To su funkcije u kontekstu klase/objekta. Na primjer:

```
class Person {

	constructor(name, email, phone) {
```

```
            this.name = name;
            this.email = email;
            this.phone = phone;
        }
        displayEmail() {
            console.log(this.email);
        }
}
```

Primjer upotrebe metode:

```
const person = new Person();
person.displayName();
```

Klase mogu naslediti drugu klasu. Na primjer:

```
class Employee extends Person {
        constructor(name, email, phone, type, position, salary) {
            super(name, email, phone);
            this.type = type;
            this.position = position;
            this.salary = salary;
        }
}
```

Nasledjivanje nam omogucava da dijelimo polja i metode izmedju odredjenih klasa.

Enkapsulacija je ideja “zatvaranja” direktnog pristupa i izmjena polja u objektu. Umjesto toga, koriste se getter i setter metode upotrebom kljucnih rijeci get i set.

Jasavscript ne podrzava potpunu enkapsulaciju po dizajnu, sva polja su tipa public. Posljednja verzija JS-a doduse napravila je pomak po tom pitanju, tako da sada mozemo imati privatna polja, a ova se oznacavaju upotrebom hash karaktera #.

```
class Person {

        constructor(name) {
            this.name = name;
        }

        get personName() {
            return this.name;
        }
```

```
    set personName(name) {
        this.name = name;
    }

}
```

Staticke metode su metode koje mozemo definisati u klasi i pozivati bez instanciranja objekta. na primjer:

```
class Person {

    constructor(name) {
        this.name = name;
    }

    static doSomething() {
        return “I’m doing it, dough!”;
    }

}
```

Zadatak 1:
Napraviti klasu Product koja ima sledeca polja:

- name, inicijalno prazan string
- price, inicijalno 0
- stockQuantity, inicijalno nula

i metodu:

- getStockValue koja vraca vrijednost na zalihi (price * stockQuantity)

Kreirati objekat product iz gore navedene klase i proslijediti vrijednosti za name, price i stockQuantity (proizvolje).
Nakon toga ispisati vrijednost koju vraca getStockValue metoda u konzoli

Zadatak 2:
Napraviti klasu Vehicle koja ima sledeca polja:

- brand, inicijalno prazan string
- numberOfWheels, inicijalno 1
- currentSpeed, inicijalno 0
- speedlimit, inicijalno 0

i metode:

- accelerate - koja koristi Math.random i generise broj izmedju 1 i speedLimit i njime azurira currentSpeed polje
- status - metoda koja provjerava trenutnu vrijednost currentSpeed polja i ako je ova:
    1. veca od speedLimit / 2: ispisuje u konzoli "You are going too fast, be careful!"
    2. manja ili jednaka speedLimit / 2: ispisuje u konzoli "Thanks for driving safely!"

Napraviti klasu Car koja ekstenduje Vehicle i:

- "gazi" numberOfWheels setujuci 4 kao inicijalnu vrijednost,
- ima polje tankCapacity cija je inicijalna vrijednost 55
- ima polje currentFuelQuantity koja opet koristi Math.random izmedju 0 i tankCapacity
- ima metodu getMissingQuantityFromTank koja ispisuje u konzoli: "You need x liters to fill up your tank!"

Kreirati instancu (objekat) iz klase Car i proslediti proizvoljne vrijednosti.
Pozvati sve metode koje rade ispis u konzoli.