# Zadání

------------------------

To zadání nám uvedli společně pro JS a PHP. Já mám PHP.

To mi bohužel ještě komplikuje, protože uvedené špatné/dobré části kódu jsou pro JS...

------------------------

#### Webová aplikace (JavaScript, PHP)

Naprogramuj webovou aplikaci pro evidenci pojistných událostí.

##### Minimální požadavky ke splnění

* Aplikace obsahuje správu **pojištěných** (to jsou pojištěné osoby, např. "Jan Novák"):
  + Vytvoření pojištěného
    - Evidujte jméno, příjmení, věk a telefonní číslo
  + Zobrazení seznamu všech pojištěných
* Dané entity jsou uloženy v kolekci v paměti (JS) nebo v databázi (PHP)
* Aplikaci lze pro zjednodušení vytvořit celou jen na jedné stránce
* Aplikace je naprogramována podle dobrých praktik

##### Dobré praktiky

* Oddělujte kód do samostatných tříd a souborů

JavaScript

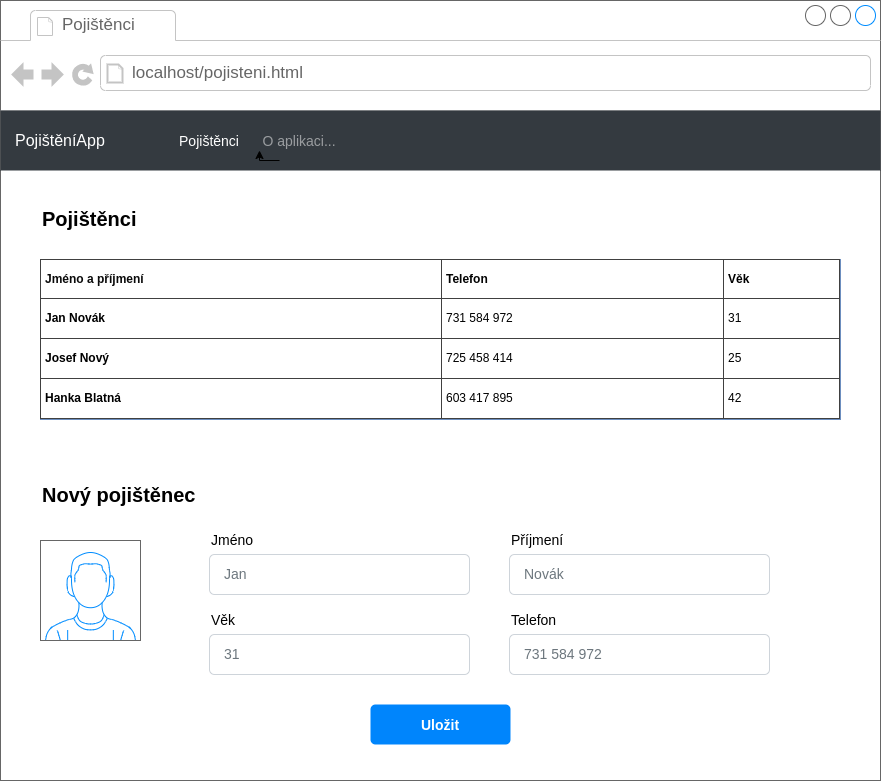
* Využívejte konstruktory pro inicializaci objektů
* toString() pro jejich výpis

PHP

* Ideálně využijte MVC architekturu

##### Ukázkové obrazovky aplikace

Pro představu si ukažme, jak může být aplikace realizována:

[](https://www.itnetwork.cz/images/5/retrainings/pojisteni_web.png)

#### kritéria hodnocení

Předpokládáme, že jsi svou aplikaci při tvorbě alespoň jednou spustil a vyzkoušel a tedy, že funguje podle funkčních požadavků výše. Proto se při hodnocení budeme zaměřovat zejména na kód, který jsi napsal a to na následujících 8 kritérií. Výčet obsahuje odkazy na dané lekce, ve kterých je problematika vysvětlena.

1. **Pojmenování identifikátorů** - Zkontroluj, zda názvy proměnných (lekce [C#](https://www.itnetwork.cz/csharp/zaklady/nejcastejsi-chyby-csharp-zacatecniku-umis-pojmenovat-promenne), [Java](https://www.itnetwork.cz/java/zaklady/nejcastejsi-chyby-java-novacku-umis-pojmenovat-promenne), [Python](https://www.itnetwork.cz/python/zaklady/nejcastejsi-chyby-python-novacku-umis-pojmenovat-promenne), [JavaScript](https://www.itnetwork.cz/javascript/zaklady/nejcastejsi-chyby-js-zacatecniku-umis-pojmenovat-promenne), [PHP](https://www.itnetwork.cz/php/zaklady/nejcastejsi-chyby-php-novacku-umis-pojmenovat-promenne)) a názvy tříd (lekce [C#](https://www.itnetwork.cz/csharp/oop/nejcastejsi-chyby-csharp-novacku-umis-pojmenovat-objekty), [Java](https://www.itnetwork.cz/java/oop/nejcastejsi-chyby-java-novacku-umis-pojmenovat-objekty), [Python](https://www.itnetwork.cz/python/oop/nejcastejsi-chyby-python-novacku-umis-pojmenovat-objekty), [JavaScript](https://www.itnetwork.cz/javascript/oop/nejcastejsi-chyby-js-novacku-umis-pojmenovat-objekty), [PHP](https://www.itnetwork.cz/php/oop/nejcastejsi-chyby-php-novacku-umis-pojmenovat-objekty)) neobsahují zkratky a jsou výstižně zvoleny
2. **Dodržení principů SRP, SoC** - Každá třída by měla mít jen jednu zodpovědnost. Zkontroluj, zda není uživatelské rozhraní v té samé třídě, která pracuje s daty. (lekce [C#](https://www.itnetwork.cz/csharp/oop/jak-spravne-rozdelit-csharp-net-aplikace-do-trid-srp-a-soc), [Java](https://www.itnetwork.cz/java/oop/jak-spravne-rozdelit-java-aplikace-do-trid-srp-a-soc), [Python](https://www.itnetwork.cz/python/oop/jak-spravne-rozdelit-python-aplikace-do-trid-srp-a-soc), [JavaScript](https://www.itnetwork.cz/javascript/oop/jak-spravne-rozdelit-javascript-aplikace-do-trid-srp-a-soc), [PHP](https://www.itnetwork.cz/php/oop/jak-spravne-rozdelit-php-aplikace-do-trid-srp-a-soc))
3. **Vhodné výběry datových typů** - Např. telefonní číslo není ve skutečnosti číslo, protože může obsahovat předvolbu a další speciální znaky jako +. Ukládáš jej jako text?
4. **Správné formátování kódu** - Kód máš správně odsazený a má jednotný styl odpovídající konvencím daného programovacího jazyka (lekce [C#](https://www.itnetwork.cz/csharp/zaklady/nejcastejsi-chyby-csharp-zacatecniku-delas-je-take), [Java](https://www.itnetwork.cz/java/zaklady/nejcastejsi-chyby-java-zacatecniku-delas-je-take), [Python](https://www.itnetwork.cz/python/zaklady/nejcastejsi-chyby-python-zacatecniku-delas-je-take), [JavaScript](https://www.itnetwork.cz/javascript/zaklady/nejcastejsi-chyby-js-zacatecniku-delas-je-take), [PHP](https://www.itnetwork.cz/php/zaklady/nejcastejsi-chyby-php-zacatecniku-delas-je-take)).
5. **Minimalizace duplicitního kódu** - Ujisti se, že neopakuješ v aplikaci několikrát stejné nebo podobné úseky kódu, které by se daly vyčlenit do jedné metody, např. že pořád nekopíruješ tu samou validaci vstupu nebo opakující se výpisy. Zkontroluj absenci nekonečných cyklů (while (true) apod). (lekce [C#](https://www.itnetwork.cz/csharp/zaklady/nejcastejsi-chyby-csharp-zacatecniku-delas-je-take), [Java](https://www.itnetwork.cz/java/zaklady/nejcastejsi-chyby-java-zacatecniku-delas-je-take), [Python](https://www.itnetwork.cz/python/zaklady/nejcastejsi-chyby-python-zacatecniku-delas-je-take), [JavaScript](https://www.itnetwork.cz/javascript/zaklady/nejcastejsi-chyby-js-zacatecniku-delas-je-take), [PHP](https://www.itnetwork.cz/php/zaklady/nejcastejsi-chyby-php-zacatecniku-delas-je-take))
6. **Validace vstupních dat a ošetření neočekávaných vstupů** - Co když někdo zadá prázdné jméno? Pro studenty C#: Ideálně ošetři i chybné zadání čísel apod. ([C#](https://www.itnetwork.cz/csharp/zaklady/c-sharp-tutorial-osetreni-uzivatelskych-vstupu)).
7. **Logičnost a přiměřená složitost algoritmů** - Zkontroluj, zda postupy nejsou zbytečně složité a metody nejsou příliš dlouhé.
8. **Přítomnost a kvalita komentářů.** - Jsou tvé třídy a jejich metody odokumentované (lekce [C#](https://www.itnetwork.cz/csharp/oop/nejcastejsi-chyby-a-dobre-praktiky-tvorby-metod-v-csharp-net), [Java](https://www.itnetwork.cz/java/oop/nejcastejsi-chyby-a-dobre-praktiky-pro-tvorbu-metod-v-jave), [Python](https://www.itnetwork.cz/python/oop/nejcastejsi-chyby-a-dobre-praktiky-pro-tvorbu-metod-v-python), [JavaScript](https://www.itnetwork.cz/javascript/oop/nejcastejsi-chyby-a-dobre-praktiky-pro-tvorbu-metod-v-js), [PHP](https://www.itnetwork.cz/php/oop/nejcastejsi-chyby-a-dobre-praktiky-pro-tvorbu-metod-v-php))? Jsou na kritických místech v aplikaci komentáře?

## nejčastější chyby

Následující chyby jsou vykopírované z reálných projektů našich studentů a jsou vybrány tak, aby zastupovaly nejčastější problémy v jejich projektech. Kód je v různých programovacích jazycích, nicméně jazyky jsou si tak podobné, že by mělo být jasné, v čem chyba spočívá, i když v tomto jazyce neprogramujete.

### Nekonečné cykly

Nekonečné cykly jsou obecně známá špatná praktika, díky které je nutné kód dodatečně zkoumat, aby bylo poznat, kdy cyklus skončí, což je nepřehledné:

✗ Špatně

while True:

print("\nVyberte si akci:")

print("1 - Přidat nového pojištěného")

print("2 - Vypsat všechny pojištěné")

print("3 - Vyhledat pojistného podle jména a příjmení")

print("4 - Konec")

...

**✔ Správně**

while volba != "4":

print("\nVyberte si akci:")

print("1 - Přidat nového pojištěného")

print("2 - Vypsat všechny pojištěné")

print("3 - Vyhledat pojistného podle jména a příjmení")

print("4 - Konec")

...

Alternativně je možné použít i boolean proměnnou:

while not konec:

print("\nVyberte si akci:")

print("1 - Přidat nového pojištěného")

print("2 - Vypsat všechny pojištěné")

print("3 - Vyhledat pojistného podle jména a příjmení")

print("4 - Konec")

...

### Nedodržení SoC

Častou chybou studentů je, že komunikují s uživatelem v té samé třídě, která slouží pro práci s daty:

✗ Špatně

class EvidencePojistenych:

def \_\_init\_\_(self):

self.pojisteni = []

def vytvor\_pojisteneho(self):

jmeno = input("Zadejte jméno: ")

prijmeni = input("Zadejte příjmení: ")

vek = input("Zadejte věk: ")

telefon = input("Zadejte telefonní číslo: ")

pojisteny = Pojisteny(jmeno, prijmeni, vek, telefon)

self.pojisteni.append(pojisteny)

print("Data byla uložena. Pokračujte ve výběru akce...")

**✔ Správně**

class EvidencePojistenych:

def \_\_init\_\_(self):

self.pojisteni = []

def pridej\_pojisteneho(self, pojisteny):

self.pojisteni.append(pojisteny)

class UzivatelskeRozhrani:

def \_\_init\_\_(self):

self.evidence = EvidencePojistenych()

def vytvor\_pojisteneho(self):

jmeno = input("Zadejte jméno: ")

prijmeni = input("Zadejte příjmení: ")

vek = input("Zadejte věk: ")

telefon = input("Zadejte telefonní číslo: ")

pojisteny = Pojisteny(jmeno, prijmeni, vek, telefon)

self.evidence.pridej\_pojisteneho(pojisteny)

print("Data byla uložena. Pokračujte ve výběru akce...")

Kód výše neobsahuje validace a nesplňuje tedy plně zadání.

### Nedodržení SRP

Studentům se občas podaří vyvořit program pouze s jednou třídou, která reprezentuje pojištěnce. Ten má zároveň chování, že umí vyhledávat ostatní pojištence a vypisovat všechny, což nedává z hlediska rozdělení odpovědnosti (SRP) smysl:

✗ Špatně

class Pojistenec:

jmeno = None

prijmeni = None

vek = None

cislo = None

def \_\_init\_\_(self, jmeno, prijmeni, vek, cislo):

self.jmeno = jmeno

self.prijmeni = prijmeni

self.vek = vek

self.cislo = cislo

@staticmethod

def pridej\_pojistence(jmeno, prijmeni, vek, cislo):

cursor.execute('INSERT INTO pojistenci(jmeno, prijmeni, vek, cislo) VALUES(?, ?, ?, ?)',

(jmeno, prijmeni, vek, cislo))

return f"Přidáno {jmeno} {prijmeni} {vek} {cislo} do databáze."

**✔ Správně**

class Pojistenec:

jmeno = None

prijmeni = None

vek = None

cislo = None

def \_\_init\_\_(self, jmeno, prijmeni, vek, cislo):

self.jmeno = jmeno

self.prijmeni = prijmeni

self.vek = vek

self.cislo = cislo

class EvidencePojistencu:

def pridej\_pojistence(jmeno, prijmeni, vek, cislo):

cursor.execute('INSERT INTO pojistenci(jmeno, prijmeni, vek, cislo) VALUES(?, ?, ?, ?)',

(jmeno, prijmeni, vek, cislo))

return f"Přidáno {jmeno} {prijmeni} {vek} {cislo} do databáze."

### Špatné formátování

Začátečníci mívají problémy hlavně se správným odsazováním kódu, který je pak nepřehledný, protože není dobře vidět, která větev programu se kdy spouští:

✗ Špatně

}

printPersons() {

const table = document.getElementById("data-table");

for (let i = 0; i < this.persons.length; i++) {

const person = this.persons[i];

const row = table.insertRow();

const nameCell = row.insertCell(0);

const surnameCell = row.insertCell(1);

const telCell = row.insertCell(2);

const ageCell = row.insertCell(3);

const dateCell = row.insertCell(4);

nameCell.innerHTML = person.name;

surnameCell.innerHTML = person.surname;

telCell.innerHTML = person.telephone;

ageCell.innerHTML = person.age;

dateCell.innerHTML = person.birthdate;

}

}

**✔ Správně**

}

printPersons() {

const table = document.getElementById("data-table");

for (let i = 0; i < this.persons.length; i++) {

const person = this.persons[i];

const row = table.insertRow();

const nameCell = row.insertCell(0);

const surnameCell = row.insertCell(1);

const telCell = row.insertCell(2);

const ageCell = row.insertCell(3);

const dateCell = row.insertCell(4);

nameCell.innerHTML = person.name;

surnameCell.innerHTML = person.surname;

telCell.innerHTML = person.telephone;

ageCell.innerHTML = person.age;

dateCell.innerHTML = person.birthdate;

}

}