**Харківський національний економічний університет**

**імені Семена Кузнеця**

**ЗВІТ**

**З ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ № 13**

**за дисципліною: *“*Технології програмування**”

**на тему: “Реалізація концепції ООП”**

**Варіант: 4**

**Виконав: студент факультету Інформаційних технологій**

**2 курсу, спец. Кібербезпека,**

**групи 6.04.125.010.21.2**

**Бойко Вадим Віталійович**

**Перевірив:**

**Венгріна Олена Сергіївна**

**ХНЕУ ім. С. Кузнеця**

**2023**

**Мета:** набуття навичок обробки виняткових ситуацій стандартними засобами мови програмування, а також створення та використання власних реалізацій винятків.

**Завданя:** Відповідно до варіанту завдання розробити методи для створення об’єктів відповідних класів із обробкою можливих виняткових ситуацій. При вводі даних користувачем використати стандартні винятки. Розробити власний клас винятків відповідно до варіанту

Варінт 4

Розробити класи Книга, Читач, Бібліотека, Читацький квиток. Реалізувати процес роботи бібліотеки. Читач може створити заявку отримання книги. Книга може бути видана, якщо вона наявна в бібліотеці. При видачі книги читачу інформація заноситься читацького квитка. При додаванні визначається дата, коли книгу треба повернути. Якщо читач спробує оформити заявку на оформлення книги, що немає в поточний момент в наявності в бібліотеці (книга(и) у інших читачів), то генерується виняток. Обробник винятку вказує на помилку й пропонує обрати іншу книгу для замовлення.

Я написав настуіпне:

*import* time  
  
  
*class* Book:  
 book\_id = 0  
  
 *def* \_\_init\_\_(self, name = ''):  
 Book.book\_id += 1  
 self.id = f'BOOK-{Book.book\_id}'  
 self.name = name  
  
 *def* \_\_str\_\_(self):  
 *return* f'Книга: {self.name}'  
  
 *def* get\_id(self):  
 *return* self.id  
  
 *def* get\_name(self):  
 *return* self.name  
  
  
*class* ReaderTicket:  
 *def* \_\_init\_\_(self, owner\_id = ''):  
 self.owner\_id = owner\_id  
 self.rented\_books = []  
  
 *def* \_\_str\_\_(self):  
 *return* f'ID відвідувача: {self.owner\_id}'  
  
 *def* add\_book(self, book = Book()):  
 self.rented\_books.append({  
 'book\_id': book.get\_id(),  
 'when\_rented': time.ctime(time.time()),  
 'should\_returned': time.ctime(time.time() + 60 \* 60 \* 24 \* 10)  
 })  
 print(f'запис у книжному білеті {self.rented\_books}')  
  
 *def* get\_rented\_books(self):  
 *return* list(self.rented\_books)  
  
 *def* get\_owner\_id(self):  
 *return* self.owner\_id  
  
  
*class* Reader:  
 reader\_id = 0  
  
 *def* \_\_init\_\_(self, name = ''):  
 Reader.reader\_id += 1  
 self.id = f'READER-{Reader.reader\_id}'  
 self.name = name  
 self.reader\_ticket = ReaderTicket(self.id)  
  
 *def* \_\_str\_\_(self):  
 *return* self.name  
  
 *def* get\_id(self):  
 *return* self.id  
  
 *def* take\_book(self, book = Book()):  
 self.reader\_ticket.add\_book(book)  
  
 *def* get\_name(self):  
 *return* self.name  
  
 *def* get\_reader\_ticket(self):  
 *return* self.reader\_ticket  
  
  
*class* Library:  
 exception\_book\_does\_not\_exist = Exception('Даної книги нажаль немає')  
  
 *def* \_\_init\_\_(self, name='', books = *None*, rented\_books = *None*):  
 *if* books *is None*:  
 books = []  
 *if* rented\_books *is None*:  
 rented\_books = []  
 self.books = books  
 self.allowed\_books = list(books)  
 self.rented\_books = rented\_books  
 self.name = name  
  
 *def* \_\_str\_\_(self):  
 *return* f'Бібліотека {self.name} вітає вас'  
  
 *def* create\_request(self, reader = Reader(), book = Book()):  
 *if* book *not in* self.allowed\_books:  
 *raise* self.exception\_book\_does\_not\_exist  
 self.allowed\_books.remove(book)  
 reader.take\_book(book)  
 print('Гарного читання')  
  
 *def* get\_books(self):  
 *return* list(self.books)  
  
 *def* get\_allowed\_books(self):  
 *return* list(self.allowed\_books)  
  
 *def* get\_name(self):  
 *return* self.name  
  
  
book\_1 = Book('book 1')  
book\_2 = Book('book 2')  
book\_3 = Book('book 3')  
book\_4 = Book('book 4')  
book\_5 = Book('book 5')  
book\_6 = Book('book 6')  
book\_7 = Book('book 7')  
  
print(book\_1.get\_id())  
print(book\_1.get\_name())  
print(book\_1)  
  
print(book\_2.get\_id())  
  
reader\_1 = Reader('reader 1')  
reader\_2 = Reader('reader 2')  
reader\_3 = Reader('reader 3')  
reader\_4 = Reader('reader 4')  
reader\_5 = Reader('reader 5')  
reader\_6 = Reader('reader 6')  
reader\_7 = Reader('reader 7')  
reader\_8 = Reader('reader 8')  
  
print(reader\_1.get\_id())  
print(reader\_1.get\_reader\_ticket())  
print(reader\_1.get\_name())  
print(reader\_1)  
  
print(reader\_2.get\_id())  
  
library = Library('library 1', [book\_1, book\_2, book\_3, book\_4, book\_5, book\_6, book\_7])  
  
print(library.get\_name())  
print(library.get\_books())  
print(library.get\_allowed\_books())  
print(library)  
  
  
  
*try*:  
 library.create\_request(reader\_1, book\_1)  
*except* Exception:  
 print('Даної книги нажаль немає')  
  
*try*:  
 library.create\_request(reader\_2, book\_2)  
*except* Exception:  
 print('Даної книги нажаль немає')  
  
*try*:  
 library.create\_request(reader\_3, book\_3)  
*except* Exception:  
 print('Даної книги нажаль немає')  
  
*try*:  
 library.create\_request(reader\_4, book\_4)  
*except* Exception:  
 print('Даної книги нажаль немає')  
  
*try*:  
 library.create\_request(reader\_4, book\_4)  
*except* Exception:  
 print('Даної книги нажаль немає')  
  
*try*:  
 library.create\_request(reader\_4, book\_4)  
*except* Exception:  
 print('Даної книги нажаль немає')  
  
*try*:  
 library.create\_request(reader\_5, book\_5)  
*except* Exception:  
 print('Даної книги нажаль немає')  
  
*try*:  
 library.create\_request(reader\_6, book\_6)  
*except* Exception:  
 print('Даної книги нажаль немає')  
  
*try*:  
 library.create\_request(reader\_7, book\_7)  
*except* Exception:  
 print('Даної книги нажаль немає')  
  
*try*:  
 library.create\_request(reader\_8, book\_1)  
*except* Exception:  
 print('Даної книги нажаль немає')

Перевірка

