Advanced Object-Oriented Programming

CPT204 – Lab 6
Erick Purwanto

Xi'an Jiaotong-Liverpool University
西交利物浦大学

CPT204 Advanced Object-Oriented Programming

Lab 6

Exception, Linked List 3, Deque 2

Welcome !

- Welcome to Lab 6 !
  - We are going to practice exception and equip LLDeque in Lab 5 with a Copy Constructor and methods that throw and catch an exception

- You will find in this lab
  1. Lab Exercise 6.1, 6.2 and 6.3, and their hints
  2. Exercise 6.1, 6.2 and 6.3

- Download **lab6** zip files from Learning Mall
- Don't forget to import the **lab6** files and the library into an IntelliJ project
  - Read **lab1** again for reference

Lab Exercise 6.1 Vehicle CONSTRUCTOR

- Complete the constructor public **Vehicle**(String type, int numWheels).
- It initializes the instance variables *type* and *numWheels*.
- Additionally, it must be illegal to construct a Vehicle of type "truck" with less than 4 wheels, or a Vehicle of type "motorcycle" with any number of wheels except 2.
    - Set the constructor so that an Illegal Argument Exception is thrown if the illegal arguments are detected.

Lab Exercise 6.2 Vehicle TEST CONSTRUCTOR

- Complete the method String **testConstructor**(String type, int numWheels).
- It tests the Vehicle constructor using the exception handling mechanism to determine whether the constructor completed normally, or an illegal argument exception thrown by the constructor in Lab Exercise 6.1 occured.
- It returns "Vehicle constructed" when the constructor successfully creates a Vehicle object,
  and returns "Illegal number of wheels" when that exception happens.
- You ***must use try-catch*** in your code.

Test Case for Lab Exercise 6.1 and Lab Exercise 6.2

- Test case 1:

```
Vehicle.testConstructor("car", 4);     →   "Vehicle constructed"
Vehicle.testConstructor("truck", 3);   →   "Illegal number of wheels"
```

WARNING: Hints to the exercise on the next slide

Please try to solve the exercise by yourself first...

Lab Exercise 6.1, 6.2 Vehicle CONSTRUCTOR and TEST Hints

- In constructor:
  - check if the type is a truck and the #wheels is less than four or the type is a motorcycle and the #wheels is not two, then throw an IllegalArgumentException object
  - initialize the two instance variables
- In test of the constructor:
  - in try:
    - pass the parameter and create a Vehicle object
    - return the first string
  - in catch of the IllegalArgumentException object ex:
    - return the second string

Lab Exercise 6.3 Vehicle COPY CONSTRUCTOR

- Complete the copy constructor public **Vehicle**(Vehicle other).
- It creates a deep copy of other.
- Test case:

```
Vehicle v1 = new Vehicle("Type A", 2);
Vehicle v2 = new Vehicle(v1);
System.out.println(v1.getType());       →      "Type A"
System.out.println(v2.getType());       →      "Type A"

v2.setType("Type B");
System.out.println(v1.getType());       →      "Type A"
System.out.println(v2.getType());       →      "Type B"
```

WARNING: Hints to the exercise on the next slide

Please try to solve the exercise by yourself first...

Lab Exercise 6.3 Vehicle COPY CONSTRUCTOR Hints

- In the copy constructor:
  - set both instance variables of the object to the respective instance variables of other
  - access the instance variables of other with dot operator

Exercise 6.1 LLDeque COPY CONSTRUCTOR

- Complete the copy constructor public **LLDeque**(LLDeque<T> other).
- It creates a deep copy of other.
- Test case 1:

```
LLDeque<String> deque = new LLDeque<>();
deque.addFirst("a");
LLDeque<String> copyDeque = new LLDeque<>(deque);
deque.addFirst("x");
copyDeque.addFirst("y");
deque.iterGet(0);                  →       "x"
deque.iterGet(1);                  →       "a"
copyDeque.iterGet(0);              →       "y"
copyDeque.iterGet(1);              →       "a"
```

Test Case for Exercise 6.2 and Exercise 6.3

- Test case 1:

```
LLDeque<String> deque = new LLDeque<>();
deque.addFirst("c");
deque.addLegalFirst("b", "x");
deque.addLegalFirst(null, "a");
deque.iterGet(0);                              →      "a"
deque.iterGet(1);                              →      "b"
deque.iterGet(2);                              →      "c"
```

Exercise 6.2 LLDeque ADD NOT NULL TO FRONT

- Complete the method void **addFirst**(T item).
- It adds an item of type T to the front of the deque.
- It must **not** use any loops or recursion.
- Each operation must take **constant time**, that is,
  it does not depend on the deque's size.
- Additionally, if the item is null, instead of adding it into the deque,
  reject and throw an illegal argument exception.

Exercise 6.3 LLDeque ADD LEGAL ITEM TO FRONT

- Complete the method void **addLegalFirst**(T item1, T item2).
- It adds the first item of type T to the front of the deque,
  but if item1 is an illegal item, it adds the second item instead.
- An illegal item is detected by handling an illegal argument exception,
  thrown by the method addFirst in Exercise 6.2.
- It must **not** use any loops or recursion.
- Each operation must take **constant time**, that is,
  it does not depend on the deque's size.
- You ***must not use null*** in your code.

Thank you for your attention !

- In this lab, you have learned:
  - To create a method that throws an exception, especially an unchecked exception called IllegalArgumentException
  - To create a method that handles that exception
  - To create a copy constructor doing a deep copy