Lab 11 Requirements

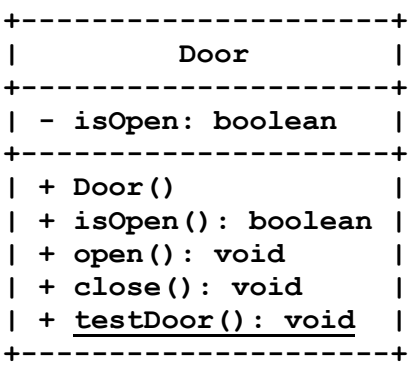
Create a new Eclipse workspace named "OOP" on the desktop of your computer. Create a Java Project named “Lab11”. Download the sample code lab11-src-for-student.zip from iSpace and unzip it. Next to import the sample code (right click the src -> Import… -> File System -> locate to the unzip folder -> import all the questions to your project) . More details, refer to file *How to Use Eclipse and submit to Autolab for student.pdf* on iSpace. Note: you must use the sample code, so AutoLab can work correctly.

Answer all questions below. Note: a question may need answers of the previous question(s), just copy paste the previous code to your new package. At the end of the lab, create a ZIP archive of all the src. Refer to previous labs requirements if you forgot how to compress files.

Extract your zip file to confirm that your zip file contains folders. Rename your src.zip to "Lab11_1234567890.zip" (replace 1234567890 with your student ID number). Upload the ZIP file onto AutoLab.

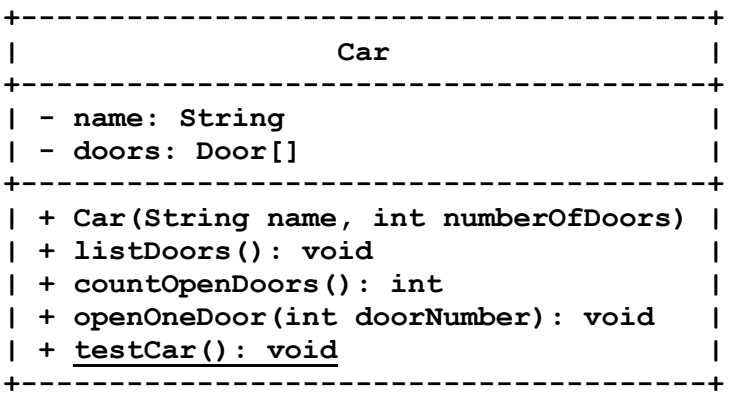
question 1 (Reviewed)

Create a class **`Door`** with the following UML specification:



where **`isOpen`** is an instance variable indicating whether the door is currently open or not. The default state of a door when it is created is to be closed.

Then create a class **`Car`** with the following UML specification:



Trying to create a car with less than one door must throw a **`BadCarException`** with the message "**`A car must have at least one door!`**". When a car is created, all its doors are closed.

The method **listDoors** prints one line of output for each door of the car. Each line starts with the name of the car followed by a message indicating whether the door is currently open or closed. For example, if a car has the name "**Tiny**" and has two doors, one which is currently open and one which is currently closed, then calling the **listDoors** method of the car object should produce the following output:

```
Tiny: door is open
Tiny: door is closed
```

The method **countOpenDoors** counts the number of doors of the car which are currently open.

The method **openOneDoor** opens a specific door, as indicated by the door's number. Doors are numbered starting from **1** (not **0**!) Trying to open a nonexistent door should throw a **BadDoorException** with the message "**Door XXX does not exist!**", where **XXX** is replaced with the number of the nonexistent door. Trying to open a door which is already open does nothing.

Use enhanced **for** loops as much as possible instead of using normal **for** loops.

Also add to your program a **Start** class with a **main** method to test your program.

question 2 (Reviewed)

Add a method **changeAllDoors** to the **Car** class that opens all the closed doors of the car and closes all the open doors of the car.

question 3 (Mandatory)

Add a method **replaceDoor** to the **Car** class that takes as argument a door number, and replaces the existing car door (as indicated by the door number) with a completely new door.

Remember that doors are numbered starting from **1** (not **0**!) Trying to replace a nonexistent door should throw a **BadDoorException** with the message "**Door XXX does not exist!**", where **XXX** is replaced with the number of the nonexistent door.

question 4 (Mandatory)

Add a method **replaceAllDoors** to the **Car** class that takes no argument and replaces all the existing car doors with completely new doors.

What happens if you try to use am enhanced **for** loop?

question 5(Mandatory)

Add a method **replaceManyDoors** to the **Car** class that takes as argument an integer **numOfDoorsToReplace** that indicates the number of existing car doors that should be replaced with completely new doors (starting from the first door in the array).

What happens if you try to replace more doors than the car has? Add a test to the **testCar** method to test this case. Make sure the test does not kill the program. (You do not need to do anything else, only test the case).

question 6(Optional)

Add a method **`expandCar`** that adds two new doors to the car. The existing doors should not be modified (the same **`Door`** objects must be used as before the car was expanded).

question 7(Optional)

Change the **`Car`** class to use an **`ArrayList`** instead of using an array. Use generics for the arraylist. Make sure all your tests still work, including the tests for exceptions.

Test Cases:

---

question 1

```
public static void testCar() {
      try {
         Car brokencar = new Car("Broken", 0);
      } catch(BadCarException ex) {
         System.out.println(ex.getMessage() == "A car must have at least one door!");
      }
      Car c = null;
      try {
         c = new Car("Biggy", 7);
      } catch(BadCarException ex) {
         System.out.println("BUG! This must never happen!");
      }
      c.listDoors();
      System.out.println(c.countOpenDoors() == 0);
      try {
         c.openOneDoor(8);
      } catch(BadDoorException ex) {
         System.out.println(ex.getMessage().equals("Door 8 does not exist!"));
      }
      try {
         c.openOneDoor(3);
      } catch(BadDoorException ex) {
         System.out.println("BUG! This must never happen!");
      }
      System.out.println(c.countOpenDoors() == 1);
}
public static void testDoor() {
    Door d = new Door();
    System.out.println(d.isOpen() == false);
    d.close();
    System.out.println(d.isOpen() == false);
    d.open();
    System.out.println(d.isOpen() == true);
}
```

question 2

```
   public static void testCar() {
      try {
         Car brokencar = new Car("Broken", 0);
      } catch(BadCarException ex) {
         System.out.println(ex.getMessage() == "A car must have at least one door!");
      }
      Car c = null;
      try {
         c = new Car("Biggy", 7);
      } catch(BadCarException ex) {
         System.out.println("BUG! This must never happen!");
      }
      c.listDoors();
      System.out.println(c.countOpenDoors() == 0);
      c.changeAllDoors();
      System.out.println(c.countOpenDoors() == 7);
      c.listDoors();
      c.changeAllDoors();
      System.out.println(c.countOpenDoors() == 0);
      try {
         c.openOneDoor(8);
      } catch(BadDoorException ex) {
         System.out.println(ex.getMessage().equals("Door 8 does not exist!"));
      }
      try {
         c.openOneDoor(3);
      } catch(BadDoorException ex) {
         System.out.println("BUG! This must never happen!");
      }
      System.out.println(c.countOpenDoors() == 1);
      c.changeAllDoors();
      System.out.println(c.countOpenDoors() == 6);
      c.listDoors();
      c.changeAllDoors();
      System.out.println(c.countOpenDoors() == 1);
      c.listDoors();
}
public static void testDoor() {
    Door d = new Door();
    System.out.println(d.isOpen() == false);
    d.close();
    System.out.println(d.isOpen() == false);
    d.open();
    System.out.println(d.isOpen() == true);
}
```

question 3

```
   public static void testCar() {
      try {
         Car brokencar = new Car("Broken", 0);
      } catch(BadCarException ex) {
         System.out.println(ex.getMessage() == "A car must have at least one door!");
      }
      Car c = null;
      try {
         c = new Car("Biggy", 7);
      } catch(BadCarException ex) {
         System.out.println("BUG! This must never happen!");
      }
```

```
      c.listDoors();
      System.out.println(c.countOpenDoors() == 0);
      c.changeAllDoors();
      System.out.println(c.countOpenDoors() == 7);
      c.listDoors();
      c.changeAllDoors();
      System.out.println(c.countOpenDoors() == 0);
      try {
         c.openOneDoor(8);
      } catch(BadDoorException ex) {
         System.out.println(ex.getMessage().equals("Door 8 does not exist!"));
      }
      try {
         c.openOneDoor(3);
      } catch(BadDoorException ex) {
         System.out.println("BUG! This must never happen!");
      }
      System.out.println(c.countOpenDoors() == 1);

      c.changeAllDoors();
      System.out.println(c.countOpenDoors() == 6);
      c.listDoors();
      c.changeAllDoors();
      System.out.println(c.countOpenDoors() == 1);
      c.listDoors();
      try {
         c.replaceDoor(8);
      } catch(BadDoorException ex) {

         System.out.println(ex.getMessage().equals("Door 8 does not exist!"));
      }

      try {
         c.replaceDoor(3);
      } catch(BadDoorException ex) {
         System.out.println("BUG! This must never happen!");
      }
      System.out.println(c.countOpenDoors() == 0);

      c.listDoors();
   }
}

public static void testDoor() {
    Door d = new Door();
    System.out.println(d.isOpen() == false);
    d.close();
    System.out.println(d.isOpen() == false);
    d.open();
    System.out.println(d.isOpen() == true);
}
```

question 4

```
   public static void testCar() {
      try {
         Car brokencar = new Car("Broken", 0);
      } catch(BadCarException ex) {
         System.out.println(ex.getMessage() == "A car must have at least one door!");
      }
      Car c = null;
```

```
      try {
         c = new Car("Biggy", 7);
      } catch(BadCarException ex) {
         System.out.println("BUG! This must never happen!");
      }
      c.listDoors();
      System.out.println(c.countOpenDoors() == 0);
      c.changeAllDoors();
      System.out.println(c.countOpenDoors() == 7);
      c.listDoors();
      c.changeAllDoors();
      System.out.println(c.countOpenDoors() == 0);
      try {
         c.openOneDoor(8);
      } catch(BadDoorException ex) {
         System.out.println(ex.getMessage().equals("Door 8 does not exist!"));
      }
      try {
         c.openOneDoor(3);
      } catch(BadDoorException ex) {
         System.out.println("BUG! This must never happen!");
      }
      System.out.println(c.countOpenDoors() == 1);

      c.changeAllDoors();
      System.out.println(c.countOpenDoors() == 6);
      c.listDoors();
      c.changeAllDoors();
      System.out.println(c.countOpenDoors() == 1);
      c.listDoors();
      try {
         c.replaceDoor(8);
      } catch(BadDoorException ex) {

         System.out.println(ex.getMessage().equals("Door 8 does not exist!"));
      }

      try {
         c.replaceDoor(3);
      } catch(BadDoorException ex) {
         System.out.println("BUG! This must never happen!");
      }
      System.out.println(c.countOpenDoors() == 0);
      c.listDoors();
      c.changeAllDoors();
      System.out.println(c.countOpenDoors() == 7);
      c.listDoors();
      c.replaceAllDoors();
      System.out.println(c.countOpenDoors() == 0);
      c.listDoors();
   }
public static void testDoor() {
     Door d = new Door();
     System.out.println(d.isOpen() == false);
     d.close();
     System.out.println(d.isOpen() == false);
     d.open();
     System.out.println(d.isOpen() == true);
}
```

question 5

```
public static void testCar() {
   try {
      Car brokencar = new Car("Broken", 0);
   } catch(BadCarException ex) {
      System.out.println(ex.getMessage() == "A car must have at least one door!");
   }
   Car c = null;
   try {
      c = new Car("Biggy", 7);
   } catch(BadCarException ex) {
      System.out.println("BUG! This must never happen!");
   }
   c.listDoors();
   System.out.println(c.countOpenDoors() == 0);
   c.changeAllDoors();
   System.out.println(c.countOpenDoors() == 7);
   c.listDoors();
   c.changeAllDoors();
   System.out.println(c.countOpenDoors() == 0);
   try {
      c.openOneDoor(8);
   } catch(BadDoorException ex) {
      System.out.println(ex.getMessage().equals("Door 8 does not exist!"));
   }
   try {
      c.openOneDoor(3);
   } catch(BadDoorException ex) {
      System.out.println("BUG! This must never happen!");
   }
   System.out.println(c.countOpenDoors() == 1);

   c.changeAllDoors();
   System.out.println(c.countOpenDoors() == 6);
   c.listDoors();
   c.changeAllDoors();
   System.out.println(c.countOpenDoors() == 1);
   c.listDoors();
   try {
      c.replaceDoor(8);
   } catch(BadDoorException ex) {

      System.out.println(ex.getMessage().equals("Door 8 does not exist!"));
   }

   try {
      c.replaceDoor(3);
   } catch(BadDoorException ex) {
      System.out.println("BUG! This must never happen!");
   }
   System.out.println(c.countOpenDoors() == 0);

   c.listDoors();
   c.changeAllDoors();
   System.out.println(c.countOpenDoors() == 7);
   c.listDoors();
   c.replaceAllDoors();
   System.out.println(c.countOpenDoors() == 0);
   c.listDoors();
   c.changeAllDoors();
   System.out.println(c.countOpenDoors() == 7);
```

```
        c.listDoors();
        c.replaceManyDoors(4);
        System.out.println(c.countOpenDoors() == 3);
        c.listDoors();
        try {
           c.replaceManyDoors(20);
        } catch(ArrayIndexOutOfBoundsException ex) {
          System.out.println(ex.getMessage().equals("Index 7 out of bounds for length 7"));
        }

        System.out.println(c.countOpenDoors() == 0);
        c.listDoors();
    }

public static void testDoor() {
        Door d = new Door();
        System.out.println(d.isOpen() == false);
        d.close();
        System.out.println(d.isOpen() == false);
        d.open();
        System.out.println(d.isOpen() == true);
  }
```

question 6

```
    public static void testCar() {
        try {
           Car brokencar = new Car("Broken", 0);
        } catch(BadCarException ex) {
           System.out.println(ex.getMessage() == "A car must have at least one door!");
        }
        Car c = null;
        try {
           c = new Car("Biggy", 7);
        } catch(BadCarException ex) {
           System.out.println("BUG! This must never happen!");
        }
        c.listDoors();
        System.out.println(c.countOpenDoors() == 0);
        c.changeAllDoors();
        System.out.println(c.countOpenDoors() == 7);
        c.listDoors();
        c.changeAllDoors();
        System.out.println(c.countOpenDoors() == 0);
        try {
           c.openOneDoor(8);
        } catch(BadDoorException ex) {
           System.out.println(ex.getMessage().equals("Door 8 does not exist!"));
        }
        try {
           c.openOneDoor(3);
        } catch(BadDoorException ex) {
           System.out.println("BUG! This must never happen!");
        }
        System.out.println(c.countOpenDoors() == 1);

        c.changeAllDoors();
        System.out.println(c.countOpenDoors() == 6);
        c.listDoors();
        c.changeAllDoors();
        System.out.println(c.countOpenDoors() == 1);
```

```
      c.listDoors();
      try {
         c.replaceDoor(8);
      } catch(BadDoorException ex) {

         System.out.println(ex.getMessage().equals("Door 8 does not exist!"));
      }

      try {
         c.replaceDoor(3);
      } catch(BadDoorException ex) {
         System.out.println("BUG! This must never happen!");
      }
      System.out.println(c.countOpenDoors() == 0);

      c.listDoors();
      c.changeAllDoors();
      System.out.println(c.countOpenDoors() == 7);
      c.listDoors();
      c.replaceAllDoors();
      System.out.println(c.countOpenDoors() == 0);
      c.listDoors();
      c.changeAllDoors();
      System.out.println(c.countOpenDoors() == 7);
      c.listDoors();
      c.replaceManyDoors(4);
      System.out.println(c.countOpenDoors() == 3);
      c.listDoors();
      try {
         c.replaceManyDoors(20);
      } catch(ArrayIndexOutOfBoundsException ex) {
        System.out.println(ex.getMessage().equals("Index 7 out of bounds for length 7"));
      }

      System.out.println(c.countOpenDoors() == 0);
      c.listDoors();


      c.expandCar();
      c.changeAllDoors();
      System.out.println(c.countOpenDoors() == 9);
      c.listDoors();
   }
public static void testDoor() {
      Door d = new Door();
      System.out.println(d.isOpen() == false);
      d.close();
      System.out.println(d.isOpen() == false);
      d.open();
      System.out.println(d.isOpen() == true);
}
```

question 7

```
   public static void testCar() {

      try {
         Car brokencar = new Car("Broken", 0);
      } catch(BadCarException ex) {
         System.out.println(ex.getMessage() == "A car must have at least one door!");
```

```
}
Car c = null;
try {
   c = new Car("Biggy", 7);
} catch(BadCarException ex) {
   System.out.println("BUG! This must never happen!");
}
c.listDoors();
System.out.println(c.countOpenDoors() == 0);
c.changeAllDoors();
System.out.println(c.countOpenDoors() == 7);
c.listDoors();
c.changeAllDoors();
System.out.println(c.countOpenDoors() == 0);
try {
   c.openOneDoor(8);
} catch(BadDoorException ex) {
   System.out.println(ex.getMessage().equals("Door 8 does not exist!"));
}
try {
   c.openOneDoor(3);
} catch(BadDoorException ex) {
   System.out.println("BUG! This must never happen!");
}
System.out.println(c.countOpenDoors() == 1);

c.changeAllDoors();
System.out.println(c.countOpenDoors() == 6);
c.listDoors();
c.changeAllDoors();
System.out.println(c.countOpenDoors() == 1);
c.listDoors();
try {
   c.replaceDoor(8);
} catch(BadDoorException ex) {

   System.out.println(ex.getMessage().equals("Door 8 does not exist!"));
}
try {
   c.replaceDoor(3);
} catch(BadDoorException ex) {
   System.out.println("BUG! This must never happen!");
}
System.out.println(c.countOpenDoors() == 0);
c.listDoors();
c.changeAllDoors();
System.out.println(c.countOpenDoors() == 7);
c.listDoors();
c.replaceAllDoors();
System.out.println(c.countOpenDoors() == 0);
c.listDoors();
c.changeAllDoors();
System.out.println(c.countOpenDoors() == 7);
c.listDoors();
c.replaceManyDoors(4);
System.out.println(c.countOpenDoors() == 3);
c.listDoors();
try {
   c.replaceManyDoors(20);
}catch(IndexOutOfBoundsException ex) {
  System.out.println(ex.getMessage().equals("Index 7 out of bounds for length 7"));
}
System.out.println(c.countOpenDoors() == 0);
```

```
        c.listDoors();
        c.expandCar();
        c.changeAllDoors();
        System.out.println(c.countOpenDoors() == 9);
        c.listDoors();
    }

public static void testDoor() {
        Door d = new Door();
        System.out.println(d.isOpen() == false);
        d.close();
        System.out.println(d.isOpen() == false);
        d.open();
        System.out.println(d.isOpen() == true);
}
```