CS101: Spring 2023 Semester
Professor Anasse Bari
Written by Christopher Davidson

Exception Detention

OI!

Through some twist of fate, you have found yourself in CS101 Exception Detention!

For the length of this lecture period, you are trapped in Detention.java with two other Students.

We can't talk during our academic incarceration, but we've developed a system of messaging via passing Exceptions to the front and back of the class.

There are currently three Students in CS101 Detention. While every Student has a name and GPA, not every type of Student is the same.

There are three types of Students currently in Detention.java:

1. **LazyBoy.java**: That's me, Sleepy Gary. I sit in the middle of the class, content with my GPA of 3.0. I have a **Student bestFriend**, and a **String storedNote**. I've gotten through CS101 without handling or creating any new Exceptions. See, if you want a message passed to the back of the class, just tell me to **getMessage(String message).** When the teacher isn't looking, I'll **quickPass()** my storedNote to my bestFriend, a TroubleMaker named Jackie Daytona.

   ```
   public void quickPass() throws Exception
   ```

   Jackie has been my **bestFriend** ever since someone called `setBFF(Student input)`) in the second grade. If Jackie has an Exception for you, I will just **throws** [sic] it to back to you.

2. **TroubleMaker.java**: Jackie Daytona, a criminal mastermind with a GPA of 1.0 who always sits at the back of the class. A Student like Jackie doesn't know a thing about exception handling, but they can create a multitude of Exceptions using the following method:

```
public void makeDistraction(String message) throws
Exception
```

Now, depending on what your `message` entails, Jackie might throw an ArithmeticException, InputMismatchException, NoSuchElementException, or an IllegalStateException. It's up to you to determine what sort of message spawns what type of Exception though. Jackie is smart enough to extract and smuggle messages from Exceptions as well, which I'm sure you know how to do, right?

3. **GoodProgrammer.java**: Finally, there's you, a **GoodProgrammer** with a GPA of 4.0. You have a string **secretMessage** that you want to pass to the back of the class, to the TroubleMaker, Jackie Daytona. To do so, you would have to first give that message to me, the proverbial middleman in this scenario, a **LazyBoy** known as Sleepy Gary.

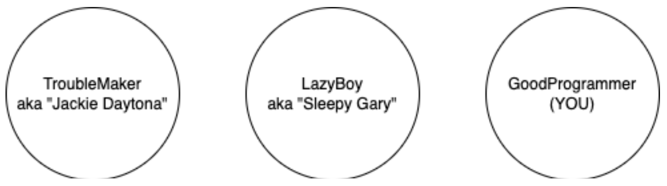
   You can use passNote(String message, Student target) to get the chain of messaging started:

   ```
   public void passNote(String message, Student
   target)
   ```

   Since you're such a GoodProgrammer™, you have always handled Exceptions whenever they are thrown at you. You know how to catch specific types, and you even have the wherewithal to print out the messages stashed within each Exception. Ergo, when you start to passNote(message, Student), you're going to want to be ready to catch any Exceptions that may come your way.

Did you get all of that? No? Okay, let's recap:

1. You are in **Detention.java**
2. There are three **Students**: The LazyBoy known as "Sleepy Gary," a TroubleMaker with the alias of "Jackie Daytona" and you, a GoodProgrammer.
3. You are in the front of the class. Sleepy Gary is immediately behind you. Jackie Daytona is behind Sleepy Gary.



(Front of Class)

4. You want to get a note all the way to Jackie Daytona in get them to make a distraction in the form of an Exception. To do so, you will **passNote()** to Sleepy Gary, who will store the note using **getMessage(String message).** Sleepy Gary will then **quickPass()** the stored note to his bestFriend, Jackie Daytona.

5. Once Jackie receives your message, he will create a new Exception (that contains a message) depending on your message's content.

    a. Make Jackie throw a NoSuchElementException("We need more Unobtanium!"), catch it, and then print it out in the error stream using System.err.println();

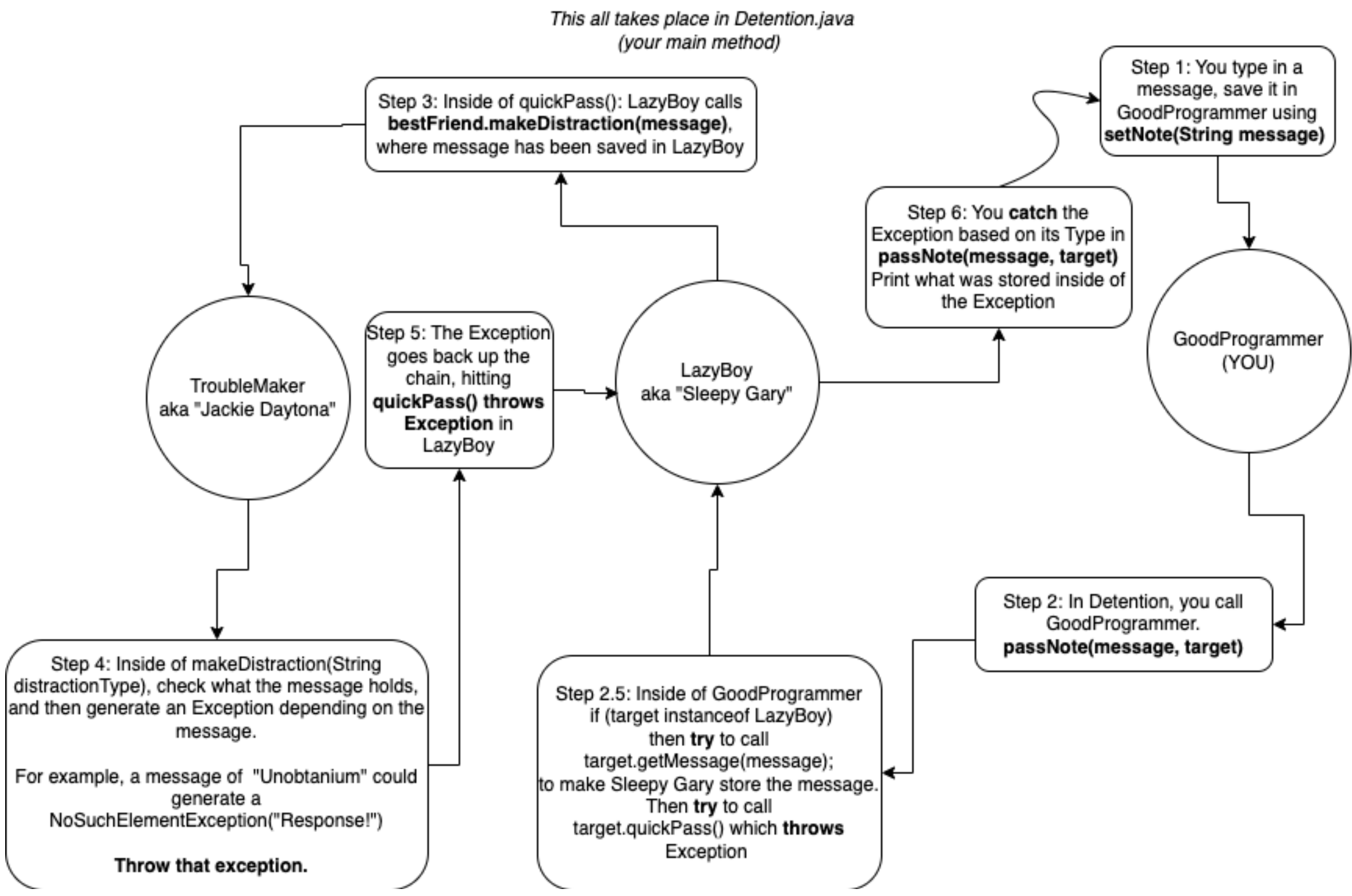
6. The LazyBoy Sleepy Gary just throws that Exception over to you, again using **quickPass().**

7. You, a GoodProgrammer better catch that Exception, and tell us what it says!

8. If you've done everything right, then you'll only have to call GoodProgrammer.passNote(message, adjacentTarget) to get the message chain going.

9. Optional: Detention doesn't last forever! If you say "QUIT" then we will let you go home! Have the TroubleMaker generate an ArrayIndexOutOfBoundsException, and then have the program terminate in a CATCH statement.

Still don't get it? Here's a complicated flow chart on the next page:


This all takes place in Detention.java (your main method)
Step 1: You type in a message, save it in GoodProgrammer using **setNote(String message)**
GoodProgrammer (YOU)
Step 2: In Detention, you call GoodProgrammer. **passNote(message, target)**
Step 2.5: Inside of GoodProgrammer if (target instanceof LazyBoy) then **try** to call target.getMessage(message); to make Sleepy Gary store the message. Then **try** to call target.quickPass() which **throws** Exception
LazyBoy aka "Sleepy Gary"
Step 3: Inside of quickPass(): LazyBoy calls **bestFriend.makeDistraction(message)**, where message has been saved in LazyBoy
TroubleMaker aka "Jackie Daytona"
Step 4: Inside of makeDistraction(String distractionType), check what the message holds, and then generate an Exception depending on the message.
For example, a message of "Unobtanium" could generate a NoSuchElementException("Response!")
Throw that exception.
Step 5: The Exception goes back up the chain, hitting **quickPass() throws Exception** in LazyBoy
Step 6: You **catch** the Exception based on its Type in **passNote(message, target)** Print what was stored inside of the Exception

Note that you do not have to use (instanceOf), if target is specifically not a Student. Since GoodProgrammer, LazyBoy and TroubleMaker are all students, however, we will have to use instanceof and casting to ensure that we can call the correct methods at the correct times.