Java Objects Part 2 Notes

Team Treehouse

May 22, 2020

1 constants

- Are named *IN_CAPITALIZED_SNAKE_CASE*
- Can be done using *static* keyword
- Allows variables and methods to be exponsed without instantiation

```java
public class PezDispenser {
    public static final int MAX_PEZ = 12; // <- 1. static declared here :)
    ...
}
```

Listing 1: lesson_1/PezDispenser.java

```java
import java.io.Console;

public class Example {
    public static void main(String[] args) {
        ...
        System.out.printf("FUN FACT: There are %d PEZ allowed in every dispenser\n", PezDispenser.MAX_PEZ); // 2. <- And is used here :)
        ...
    }
}
```

Listing 2: lesson_1/Example.java

Notes:

- Files can be compiled and displayed by typing *javac Example.java && java Example* in terminal

2 Exercise 1

- Solution included in *exercise_1.java*

3 Filling the Dispenser

- *void* keyword means nothing is returned at the end of a method

```java
    public class PezDispenser {
        public void fill() { // <- This little guy here :)
            this.pezCount = MAX_PEZ;
            System.out.printf("The current count of delicious PEZ is %d\n", this.pezCount);
        }
    }

```

Listing 3: lesson_3/PezDispenser.java

```java
    import java.io.Console;

    public class Example {
        public static void main(String[] args) {
            ...
            dispenser.fill(); // <- 2. Is used like this

        }
    }

```

Listing 4: lesson_3/Example.java

Notes:

- Files can be compiled and displayed by typing *javac Example.java && java Example* in terminal
- Always start with private methods, and turn to public when needed.

4 Exercise 2

- Solution included in *exercise_2.java*

5 Abstraction at Play

- *Golden Rule* Don't make users understand object internally
  - Simple questions such as 'is it empty?' is sufficent

```java
public class PezDispenser {
    public boolean isEmpty() { // <- This little guy here :)
        return this.pezCount == 0;
    }

    ...
}

```

Listing 5: lesson_5/PezDispenser.java

```java
import java.io.Console;

public class Example {
    public static void main(String[] args) {
        ...
        if (dispenser.isEmpty()) {
            System.out.printf("Dispenser is empty"); // <- 2. with this little fellow here
        }

        ...
        if (!dispenser.isEmpty()) {
            System.out.printf("Dispenser is full\n"); // <- 3. and this guy as well
        }

    }
}

```

Listing 6: lesson_5/Example.java

<u>Notes:</u>

- Files can be compiled and displayed by typing *javac Example.java && java Example* in terminal

6 Exercise 3

- Solution included in *exercise_3.java*

7 Incrementing and Decrementing

- *INT_VARIABLE–:* Decrements the value in variable by 1
- *INT_VARIABLE++:* Increments the value in variable by 1

```java
public class PezDispenser {
    ...
    public boolean dispense() { // <- 1. This little guy here :)
        boolean wasDispensed = false;
        if (!this.isEmpty()) {
            this.pezCount--; // <- 2. With decrement count here
            wasDispensed = true;
        }

        return wasDispensed;

    }
}

```

Listing 7: lesson_7/PezDispenser.java

```java
import java.io.Console;

public class Example {
    public static void main(String[] args) {
        ...
        while (dispenser.dispense()) {
            System.out.println("Chomp!"); // <- 3. This will print
as long as .dispensed() returns true
        }

        if (dispenser.isEmpty()) {
            System.out.println("Ate all the PEZ");
        }
    }
}

```

Listing 8: lesson_7/Example.java

```
>>> javac Example.java && java Example
We are making a new PEZ dispenser

FUN FACT: There are 12 PEZ allowed in every dispenser
Dispenser is emptyThe dispenser is Yoda
Filling the dispenser with delicious PEZ...
The current count of delicious PEZ is 12
Dispenser is full
Chomp!
```

```
Chomp!
Chomp!
Chomp!
Chomp!
Chomp!
Chomp!
Chomp!
Chomp!
Chomp!
Chomp!
Chomp!
Ate all the PEZ

```

Notes:

– Files can be compiled and displayed by typing *javac Example.java && java Example* in terminal

8 Exercise 4

- Solution included in *exercise_4.java*

9 Method Overloading

- Is a feature that llows a class to have more than one method with the same name, if their arguement lists are different.

```
public class PezDispenser {
    public void fill() { // <- 1. This little guy here
        this.fill(MAX_PEZ);
    }

    public void fill(int pezAmount) { // 2. And this little guy here :)
        this.pezCount = pezAmount;
        System.out.printf("The current count of delicious PEZ is %d\n", this.pezCount);
    }

}

```

Listing 9: lesson_9/PezDispenser.java

```java
import java.io.Console;

public class Example {
    public static void main(String[] args) {
        ...
        while (dispenser.dispense()) {
            System.out.println("Chomp!"); // <- 3. This will print
as long as .dispensed() returns true
        }

        if (dispenser.isEmpty()) {
            System.out.println("Ate all the PEZ");
        }
    }
}

```

Listing 10: lesson_9/Example.java

```
>>> javac Example.java && java Example
We are making a new PEZ dispenser

FUN FACT: There are 12 PEZ allowed in every dispenser
Dispenser is emptyThe dispenser is Yoda
Filling the dispenser with delicious PEZ...
The current count of delicious PEZ is 12
Dispenser is full
Chomp!
Chomp!
Chomp!
Chomp!
Chomp!
Chomp!
Chomp!
Chomp!
Chomp!
Chomp!
Chomp!
Chomp!
Ate all the PEZ
The current count of delicious PEZ is 2
Chomp!!
Chomp!!

```

Listing 11: Terminal

10 Exercise 5

- Solution included in *exercise_5.java*

11 Exceptions

- Exception can be raised by using *throw*
- Like python, *try* and *catch* exists.

```java

public class PezDispenser {
    ...
    public void fill(int pezAmount) {
        int newAmount = pezCount + pezAmount;
        if (newAmount > MAX_PEZ) { // <- 1. This little guy here
            throw new IllegalArgumentException("Too many Pez");
        }

        this.pezCount = pezAmount;
        System.out.printf("The current count of delicious PEZ is %d\n", this.pezCount);
    }

}

```

Listing 12: lesson_9/PezDispenser.java

```java
import java.io.Console;

public class Example {
    public static void main(String[] args) {

        ...

        try {
            dispenser.fill(400);
            System.out.println("This will never happen");
        } catch(IllegalArgumentException iae) {  // <- 2. causes this exception to throw :)
            System.out.println("Whoa there!!");
            System.out.printf("The error was %s\n", iae.getMessage()); // 3. <- giving this message
        }
    }
}

```

Listing 13: lesson_9/Example.java

```
>>> javac Example.java && java Example
...
Whoa there!!
The error was Too many Pez

```

Listing 14: Terminal

Notes:

- Files can be compiled and displayed by typing *javac Example.java && java Example* in terminal

12 Exercise 6

- Solution included in *exercise_6.java*

13 Exercise 7

- Solution included in *exercise_7.java*