//TODO: Write A Better Comment

Adam McNeilly - @AdamMc331

Java @java · Jun 2

Stop Writing Code Comments

#cleancoder

medium.com/@bpnorlander/s...

```
// this function sends an email
void sendEmail() {
  ...
}

// this class holds data for an employee
public class Employee {
  ...
}

/**
 * @param title The title of the CD
 * @param author The author of the CD
 * @param tracks The number of tracks on the CD
```

257 298 1K

This Is Bad Advice

It Also Doesn't Need To Be This Harsh

"When you need to write a comment, it usually means that you have failed to write code that was expressive enough. You should feel a sharp pain in your stomach every time you write a comment."

You Are Not A Failure For Writing Comments

We Need To Stop Writing *Bad* Comments

Why Do We Have Comments, Anyways?

They Provide Additional Insight

```
/**
 * There is certain functionality that we need to be consistent
 * in all WebViews of our app.
 * For some URLs, though, we need additional customization so we
 * can extend this base class accordingly.
 */
class BaseWebViewClient(...) : WebViewClient
```

They Can Tell You Why The Programmer Did Something

```
// The API returns the time in seconds
// but we need to manipulate it as milliseconds.
val timeInMillis = response.time * 1000
```

They Can Provide Documentation

```
interface AccountDAO {
    /**
     * Inserts an account into the database.
     *
     * @param[account] The account that we're inserting.
     * @return The ID of the inserted account.
     */
    fun insert(account: Account): Long
}
```

What Risks Do Comments Pose?

Changing Code Doesn't Guarantee We Change Comments

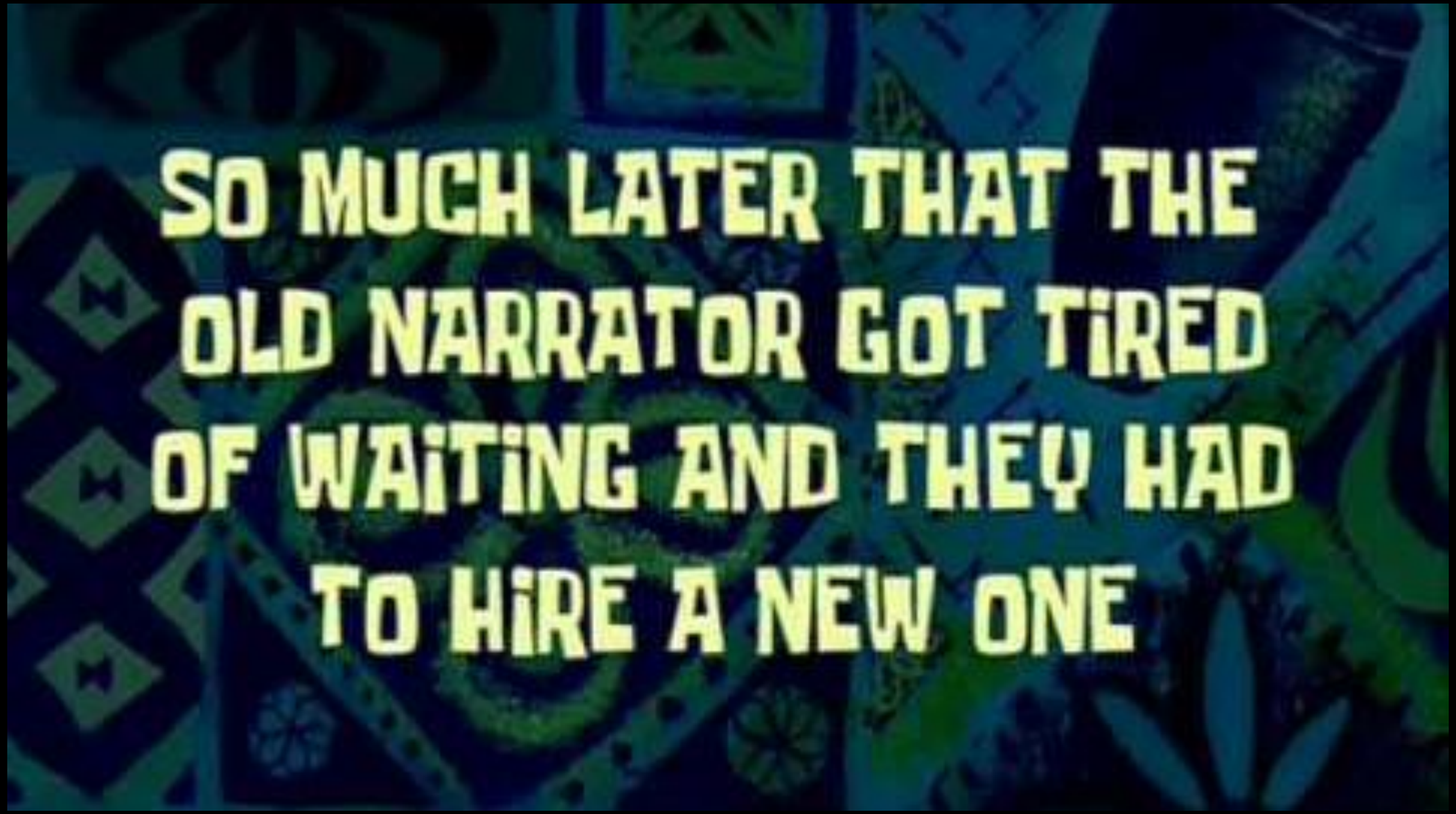
Describe Some Action

```
// We only want active users
val usersToDisplay = userList.filter { user ->
    user.isActive
}
```


THREE
WEEKS
LATER...

That Action Changed

```
// We only want active users
val usersToDisplay = userList.filter { user ->
    user.isActive && user.completedRegistration
}
```


SO MUCH LATER THAT THE
OLD NARRATOR GOT TIRED
OF WAITING AND THEY HAD
TO HIRE A NEW ONE

Who is right? 🤔

```
// We only want active users
val usersToDisplay = userList.filter { user ->
    user.isActive && user.completedRegistration
}
```

Managing Code And Comments Is Difficult

- Try to avoid them by default
- Don't avoid comments just for avoidance sake
- Ask yourself if there's some way to avoid it

Avoid Redundant Comments

The Comments Tell Me Everything The Code Does

```
interface AccountDAO {
    /**
     * Inserts an account into the database.
     *
     * @param[account] The account that we're inserting.
     * @return The ID of the inserted account.
     */
    fun insert(account: Account): Long
}
```

Remove What We Don't Need

```
interface AccountDAO {
    /**
     * @return The ID of the inserted account.
     */
    fun insert(account: Account): Long
}
```

An Exception

- If you're writing a library or public facing API, document everything
- More on this later

Change Code To Avoid Needing Comments

Sometimes We Try To Clarify Behavior

```
// Saves data to database
fun saveData() {
    // ...
}
```

We Can Write More Expressive Method Names

```
fun saveDataToDatabase() {
    // ...
}
```

Sometimes We Use Them To Break Up A Method[1]

```
fun transferMoney(fromAccount: Account, toAccount: Account, amount: Double) {
    // create withdrawal transaction and remove from fromAccount
    // ...

    // create deposit transaction and add from toAccount
    // ...
}
```

[1] This is also bad because methods should do just one thing.

We Should View This As An Opportunity To Extract Functionality

```
fun transferMoney(fromAccount: Account, toAccount: Account, amount: Double) {
    withdrawMoney(fromAccount, amount)
    depositMoney(toAccount, amount)
}
```

Now What?

- We removed any redundant comments
- We changed code to avoid comments
- But we still feel the need to clarify what we did, or we're working on a public API

How Do I Ensure The Comments I Do Write Are Helpful?

Comments Tell You *Why*, Code Tells You *What*

This Comment Only Tells Me What

```
/**
 * A list of updated questions to be replaced in our list by an interceptor.
 */
private val updatedQuestions: MutableMap<Long, Question> = HashMap()
```

We Should Clarify Why We Need This Value

```
/**
 * The `PagedList` class from Android is backed by an immutable list.
 * However, if the user answers a question locally, we want to update the display
 * without having to fetch the data from the network again.
 *
 * To do that, we keep this local cache of questions that the user has
 * answered during this app session, and later when we are building
 * the list we can override questions with one from this list, if it exists.
 * That's determined by the key of this HashMap, which is the question ID.
 */
private val updatedQuestions: MutableMap<Long, Question> = HashMap()
```

Comments With Examples Are Helpful

These Are All Redundant

```
class Pokedex {
    /**
     * Adds a pokemon to this pokedex.
     *
     * @param[name] The name of the Pokemon.
     * @param[number] The number of the Pokemon.
     */
    fun addPokemon(name: String, number: Int) {

    }
}
```

We Can Provide Examples

```
class Pokedex {
    /**
     * Adds a pokemon to this Pokedex.
     *
     * @param[name] The name of the Pokemon (Bulbasaur, Ivysaur, Venusaur).
     * @param[number] The number of the Pokemon (001, 002, 003).
     */
    fun addPokemon(name: String, number: Int) {

    }
}
```

Links To External Resources Are Helpful

For Things We Find On StackOverflow

```
/**
 * A ViewPager that cannot be swiped by the user,
 * but only controlled programatically.
 *
 * Inspiration: https://stackoverflow.com/a/9650884/3131147
 */
class NonSwipeableViewPager(
    context: Context,
    attrs: AttributeSet? = null
) : ViewPager(context, attrs) {
    // ...
}
```

For Internal Documentation

```
/**
 * Implementation of some feature that I was asked to build.
 *
 * Design/Product Spec: https://confluence.com/some/feature
 */
class SomeFeatureFragment : Fragment() {
    // ...
}
```

For Work Arounds Of Reported Issues

```
/**
 * The carousel library we use does not support a
 * specific functionality that we need. We've extended
 * this class to modify it ourselves.
 *
 * Issue reported: https://github.com/library/issues/1
 */
class MyCustomCarousel : Carousel() {
    // ...
}
```

Actionable Comments Are Helpful

//TODO: Comments

Two Options For //TODO: Comments

Option 1: Just Do It

Option 2: Link To External Issue Tracker

```
//TODO: Consolidate both of these classes
// since we only have one activity now.
// AAA-123
class MainActivity : BaseActivity() {
    // ...
}
```

Deprecation Comments Can Be Actionable

Don't Do This

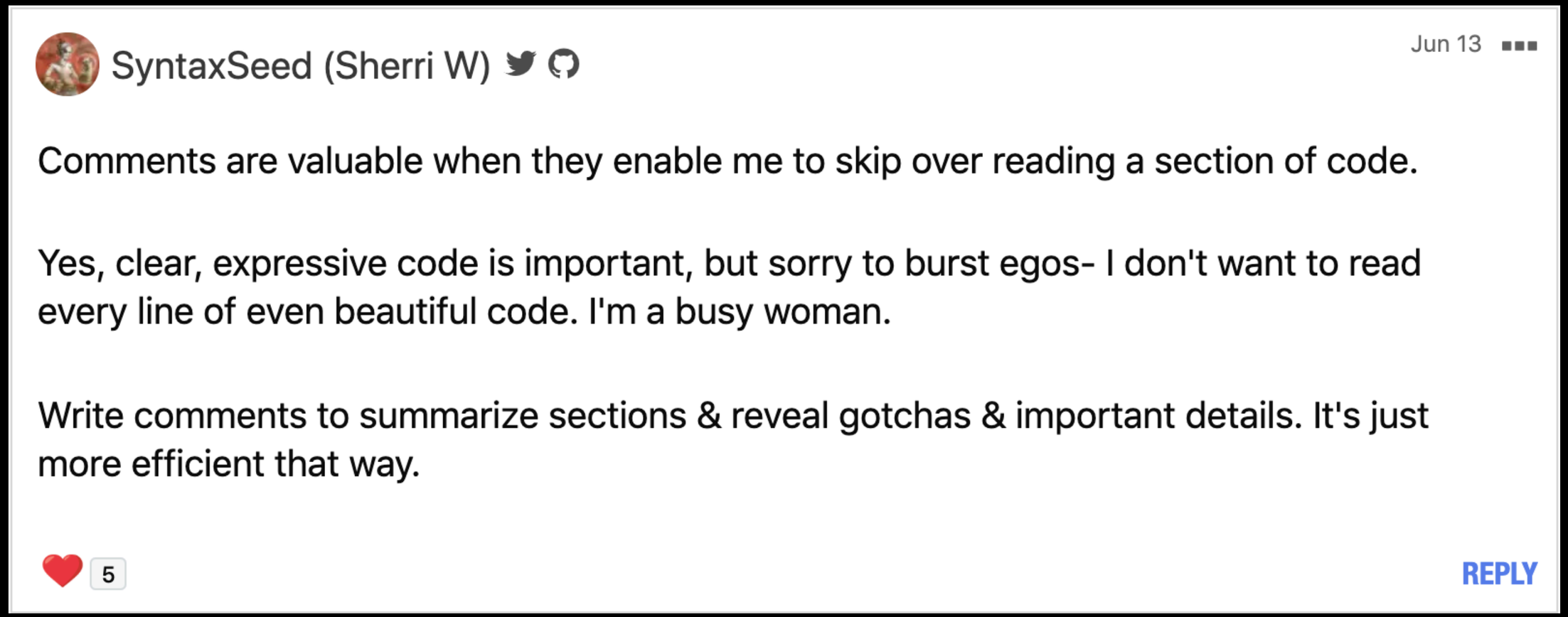
```
@Deprecated
public interface DefaultBehavior {
    // ...
}
```

Tell People What The Replacement Is

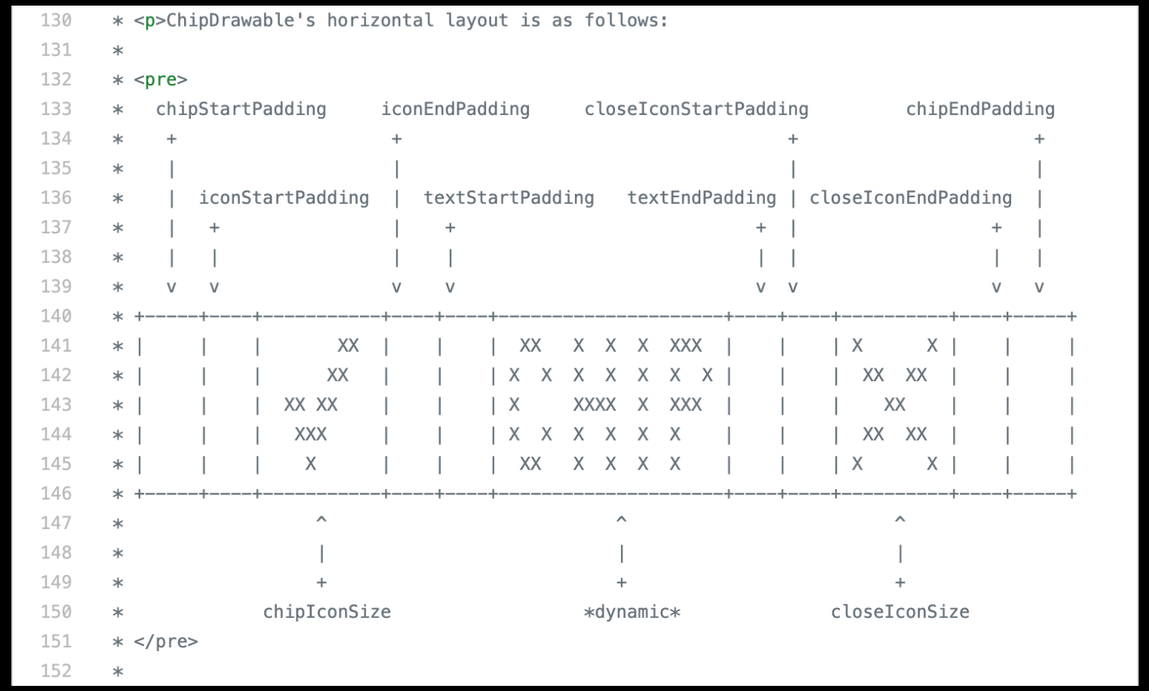
```
/**
 * @deprecated Use {@link AttachedBehavior} instead
 */
@Deprecated
public interface DefaultBehavior {
    // ...
}
```

Other General Suggestions

Try To Summarize Large Sections Of Code



ASCII Art?[2]



```
130   * <p>ChipDrawable's horizontal layout is as follows:
131   *
132   * <pre>
133   *   chipStartPadding     iconEndPadding     closeIconStartPadding         chipEndPadding
134   *    +                    +                                   +                       +
135   *    |                    |                                   |                       |
136   *    |  iconStartPadding  |  textStartPadding   textEndPadding | closeIconEndPadding   |
137   *    |   +                |    +                            +  |                   +   |
138   *    |   |                |    |                            |  |                   |   |
139   *    v   v                v    v                            v  v                   v   v
140   * +-----+----+-----------+----+----+--------------------+----+----+----------+----+-----+
141   * |     |    |       XX  |    |    |  XX   X  X  X  XXX  |    |    | X      X |    |     |
142   * |     |    |      XX   |    |    | X  X  X  X  X  X  X |    |    |  XX  XX  |    |     |
143   * |     |    |  XX XX    |    |    | X     XXXX  X  XXX  |    |    |    XX    |    |     |
144   * |     |    |   XXX     |    |    | X  X  X  X  X  X    |    |    |  XX  XX  |    |     |
145   * |     |    |    X      |    |    |  XX   X  X  X  X    |    |    | X      X |    |     |
146   * +-----+----+-----------+----+----+--------------------+----+----+----------+----+-----+
147   *                ^                         ^                      ^
148   *                |                         |                      |
149   *                +                         +                      +
150   *           chipIconSize               *dynamic*            closeIconSize
151   * </pre>
152   *
```

[2] https://github.com/material-components/material-components-android/blob/master/lib/java/com/google/android/material/chip/ChipDrawable.java#L130-L151

Reference Classes/Methods/Properties

- Helps survive refactoring of a field
- IDE may let you click into a reference

Without References

```
/**
 * Retrieves the primary Type for a Pokemon.
 */
val firstType: Type?
    get() = currentState.pokemon?.sortedTypes?.firstOrNull()
```

Refactor class name...

```
/**
 * Retrieves the primary Type for a Pokemon.
 */
val firstType: PokemonType?
    get() = currentState.pokemon?.sortedTypes?.firstOrNull()
```

With References

```
/**
 * Retrieves the primary [Type] for a [Pokemon].
 */
val firstType: Type?
    get() = currentState.pokemon?.sortedTypes?.firstOrNull()
```

Refactor class name...

```
/**
 * Retrieves the primary [PokemonType] for a [Pokemon].
 */
val firstType: PokemonType?
    get() = currentState.pokemon?.sortedTypes?.firstOrNull()
```

Be Consistent With Your Language

- When documenting methods that return booleans, try to always describe the true condition
- Don't describe the true response for some methods and the false response for others

Inconsistent Documentation

```
/**
 * @return True if the user has signed on within the last 24 hours.
 */
fun isActive(): Boolean {
    // ...
}

/**
 * @return False if the user is not a staff member for our team.
 */
fun isStaff(): Boolean {
    // ...
}
```

Consistent Documentation

```
/**
 * @return True if the user has signed on within the last 24 hours.
 */
fun isActive(): Boolean {
    // ...
}

/**
 * @return True if the user is a staff member of our team.
 */
fun isStaff(): Boolean {
    // ...
}
```

Recap

- Avoid redundant comments
- Try to refactor your code to avoid a comment
- If you have to write a comment, try your best to be helpful
  - Explain *why*
  - Provide examples
  - Be actionable
  - Leverage IDE tools
  - Be consistent

Thank You!

https://github.com/AdamMc331/TODO-DCNYC19