Ted M. Young
Java Trainer, Coach, & Live Coder

Stop Obsessing About Primitives

Your Classes Are Too Big

Get in touch: ted@tedmyoung.com
Twitter: @JitterTed
Twitch: https://JitterTed.Stream
YouTube: https://JitterTed.TV
About me: https://TED.dev/about

Source Code? Slides? Sign up at:
https://mycmt.dev/

Ted M. Young

ted@tedmyoung.com

I Can Help Your Team…

Write more Testable code with more Effective tests

Become better developers

Be more productive in **Java & Spring**

Effectively use **TDD**

Improve Their **Refactoring Skills**

Your Classes are too BIG

They Have Too Much Detailed State

... which is hard to understand

... and especially test

The Blackjack Card Game

Blackjack Game State

```
public class Game {

    private final List<Card> deck;

    private final List<Card> dealerHand = new ArrayList<>();
    private final List<Card> playerHand = new ArrayList<>();

    private boolean initialCardsDealt;
    private boolean playerDone;
    private boolean dealerDone;

    // rest of class
}
```


Blackjack Game State

```
public class Game {

    private final List<Card> deck;

    private final List<Card> dealerHand = new ArrayList<>();
    private final List<Card> playerHand = new ArrayList<>();

    private boolean initialCardsDealt;
    private boolean playerDone;
    private boolean dealerDone;

    // rest of class
}
```


The Hunt for Primitives

What Are They and Where Do We Find Them?

Primitives: What Are They?

int, long, float, char	boolean	String	List, Map, Set, Arrays []
Usual Suspects	State Machine in hiding?	leads to Stringly-Typed code	Rich in Behavior

```
int ticketQuantity
long eventVenueId
float surchargePercent
```

```
boolean isSubmitted
boolean isValid
boolean isApproved
boolean isCompleted
```

```
private void handle(String command) {
    if (command.toLowerCase().startsWith("h")) {
        game.playerHits();
    } else if (command.toLowerCase().startsWith("s")) {
        game.playerStands();
    }
}
```

```
boolean initialCardsDealt
boolean playerDone
boolean dealerDone
```


Primitives: What Are They?

int, long, float, char	boolean	String	List, Map, Set, Arrays []
Suspects	in hiding?	to Stringly-Typed code	Behavior

Domain-Free Types

Primitives: Where Are They?

- ✅ Instance variables (fields)

```
private String username;
```

- ✅ Method parameters

```
public void changeUsernameTo(String username) {...}
```

- ✅ Return values

```
public String username() { return username; }
```

- 🚫 *Not local variables*

Primitives: What's Wrong?

- Mixed responsibilities
- Dispersed logic
- Unclear interactions between fields
- Lacks Units and Range
- Hard to test

Mixed Detailed Behavior

Class Handles Technical Details for Everything

```
public void initialDeal() {...}
public void playerHits() {...}
public void playerStands() {...}
public boolean isPlayerDone() {...}
public void dealerTurn() {...}
public List<Card> playerHand() {...}
public List<Card> dealerHand() {...}
public int playerHandValue() {...}
public int dealerHandValue() {...}
public GameOutcome determineOutcome() {...}
public boolean isBusted(List<Card> hand) {...}
public boolean hasBlackjack(List<Card> hand) {...}
private List<Card> createShuffledDeck() {...}
private void dealRoundOfCards() {...}
private int handValue(List<Card> hand) {...}
```


```
public void initialDeal() {...}
public void playerHits() {...}
public void playerStands() {...}
public boolean isPlayerDone() {...}
public void dealerTurn() {...}
public List<Card> playerHand() {...}
public List<Card> dealerHand() {...}
public int playerHandValue() {...}
public int dealerHandValue() {...}
public GameOutcome determineOutcome() {...}
public boolean isBusted(List<Card> hand) {...}
public boolean hasBlackjack(List<Card> hand) {...}
private List<Card> createShuffledDeck() {...}
private void dealRoundOfCards() {...}
private int handValue(List<Card> hand) {...}
```


Too many Primitives and Behavior in a Single Class

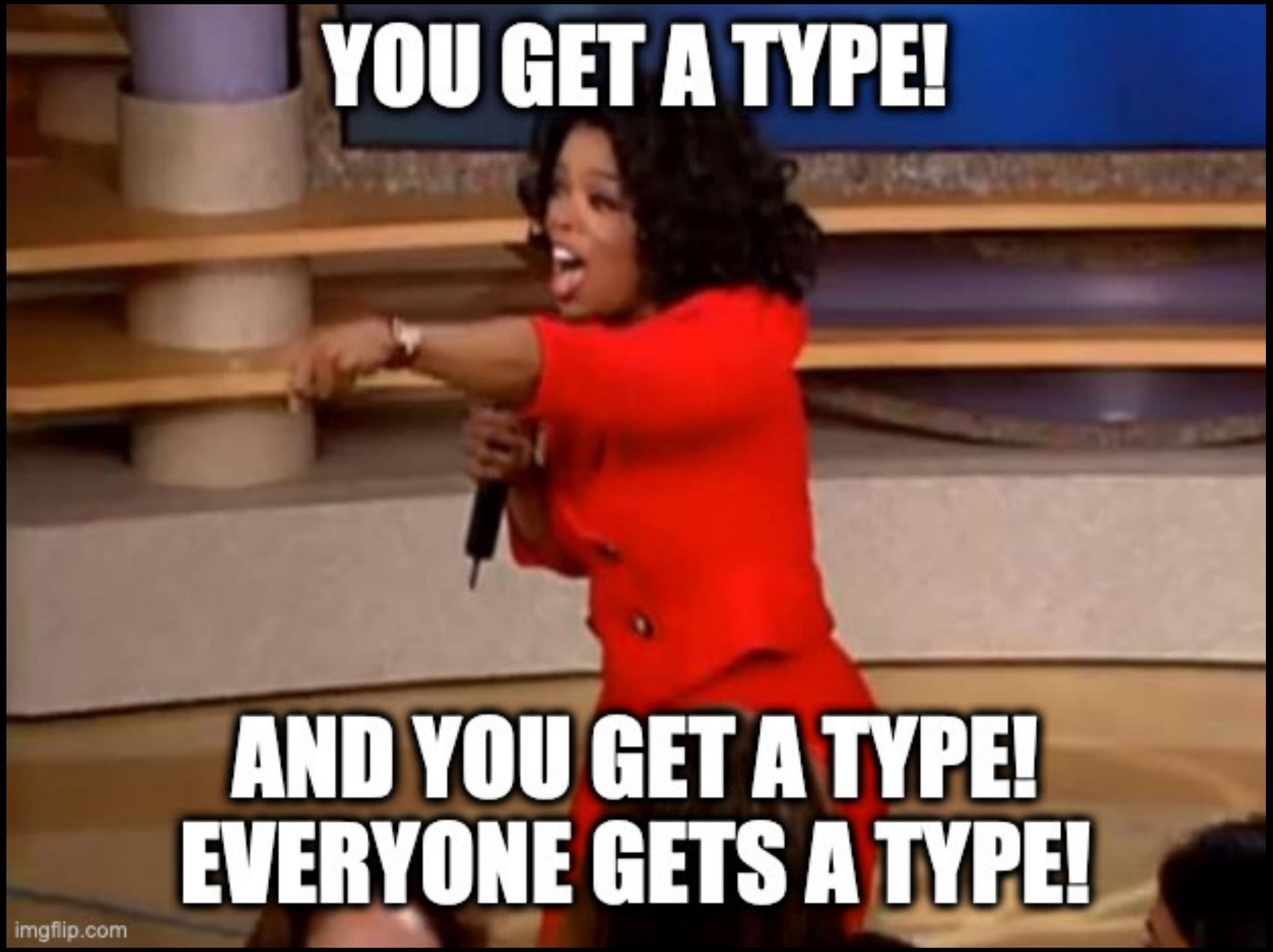
Primitive Obsession Defined

Primitive Obsession Checklist

"Domain-Free" Type	Used in Behavior, Logic, or Loop	Containing Class Does Other Things
✓	✓	✓

Fixing Primitive Obsession

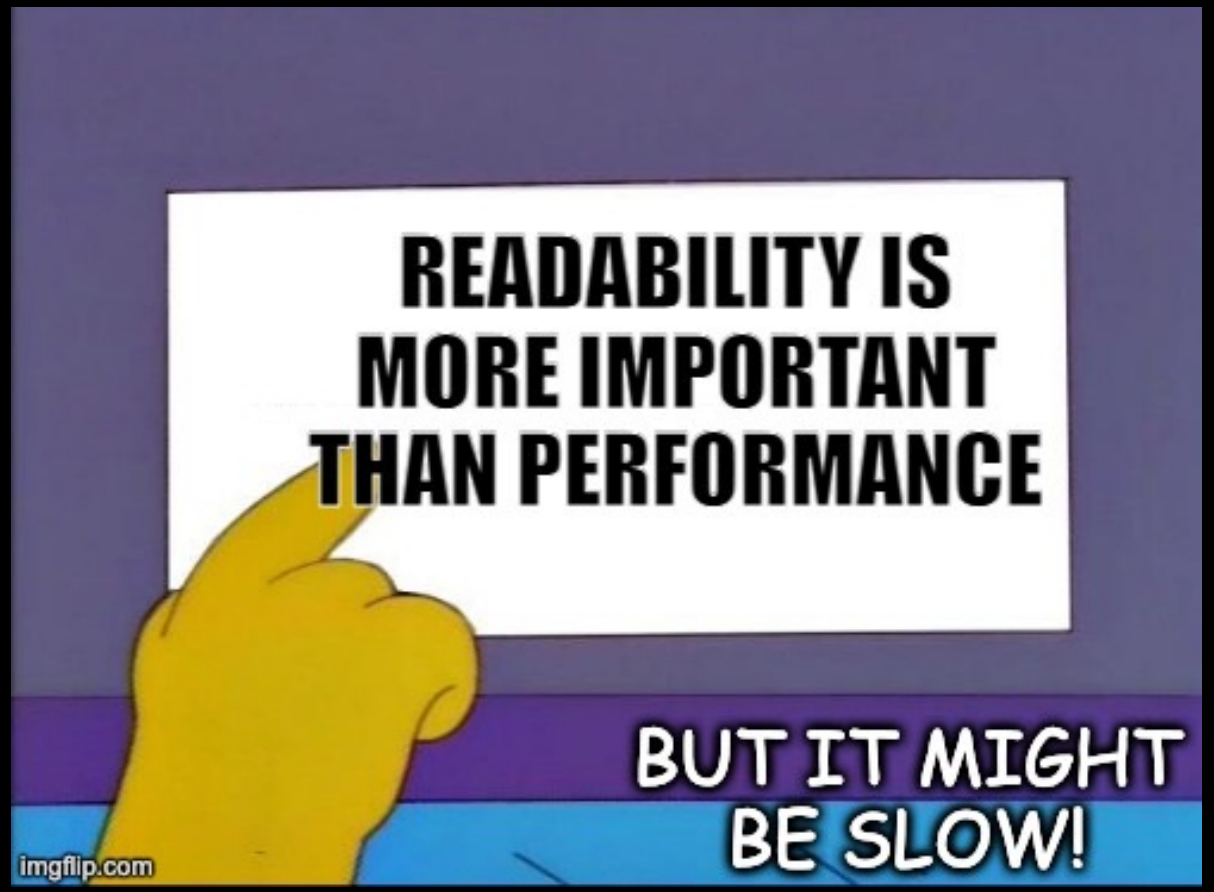
Create New Types


YOU GET A TYPE!
AND YOU GET A TYPE!
EVERYONE GETS A TYPE!
imgflip.com

Readability > Performance

We often over-emphasize performance needs


DON'T MAKE ME TAP THE SIGN.


READABILITY IS MORE IMPORTANT THAN PERFORMANCE
BUT IT MIGHT BE SLOW!
imgflip.com

Many More Smaller Types...

- Deposit Amount
- Withdrawal Amount
- Account Balance
- Discount
- Email
- Username
- Ticket Quantity
- Inventory Level
- Weight?
  - Shipping Weight
  - Coffee Bean Weight
- Temperature?
  - Body Temperature
  - Oven Temperature

Enum

Usage: Limited Options Known at Coding Time

Enum Examples

- Playing Card Suit or Rank
  - Hearts, Clubs; Ace, 2, 3..., Queen, King
- Game Outcomes
  - Beat Dealer, Lost to Dealer, Went “Bust”, Won Blackjack
- Card Face
  - Up or Down
- Option Contract Type
  - Put or Call

```java
public class Card {
  private final String suit;
  private final String rank;

  public Card(String suit, String rank) {
    this.suit = suit;
    this.rank = rank;
  }

  public int rankValue() {
    if ("JQK".contains(rank)) {
      return 10;
    } else if (rank.equals("A")) {
      return 1;
    } else {
      return Integer.parseInt(rank);
    }
  }
```

Enum: Turn Logic into Constant

```java
public enum Rank {
  ACE(1, "A"), TWO(2, "2"), THREE(3, "3"), FOUR(4, "4"), FIVE(5, "5"),
  SIX(6, "6"), SEVEN(7, "7"), EIGHT(8, "8"), NINE(9, "9"),
  TEN(10, "10"), JACK(10, "J"), QUEEN(10, "Q"), KING(10, "K");

  private final int value;
  private final String display;

  Rank(int value, String display) {
    this.value = value;
    this.display = display;
  }

  public int value() {
    return value;
  }

  public String display() {
    return display;
  }
}
```

Enum: Turn Logic into Constant

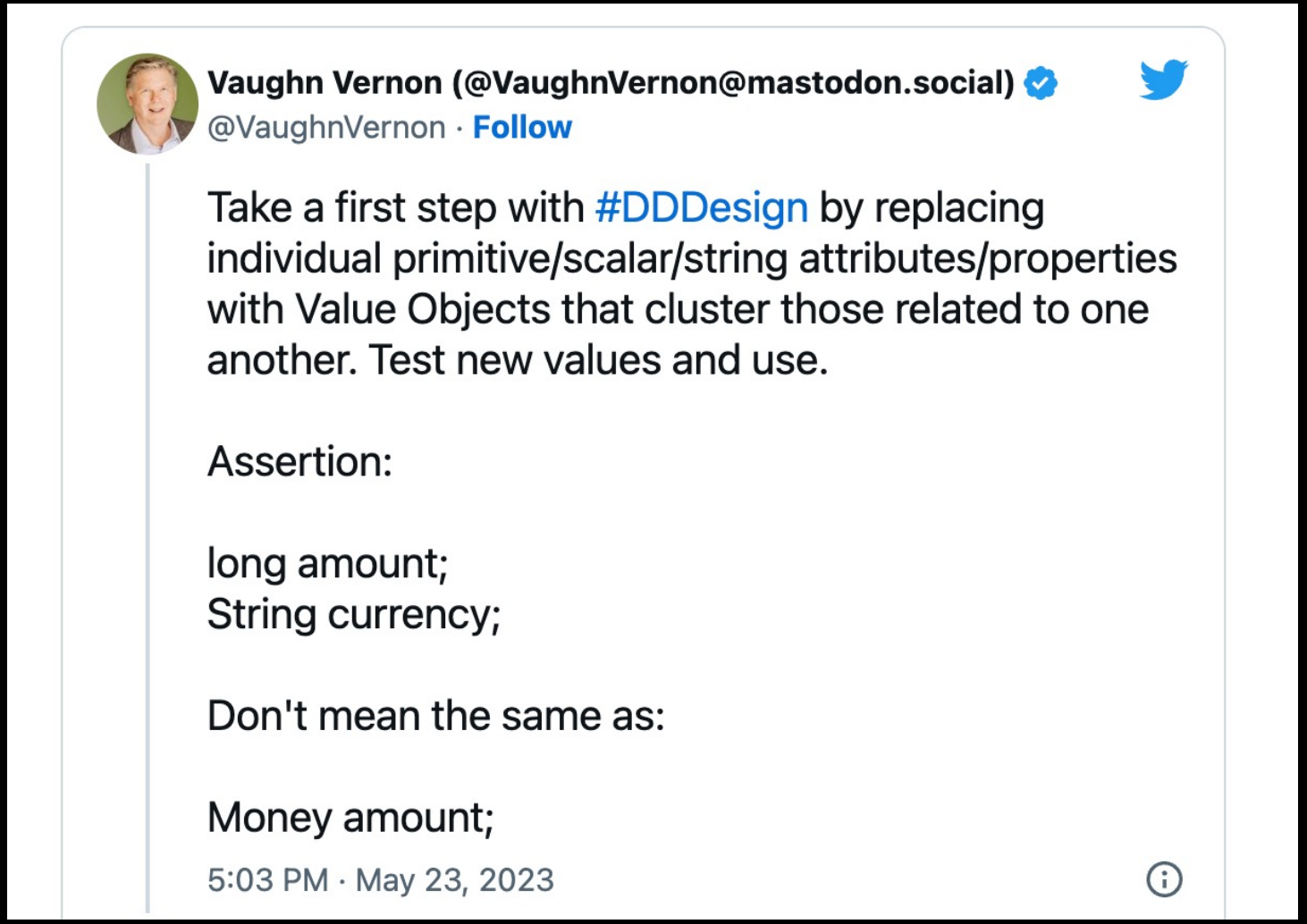
Value Object

Usage: Many Options, May Change Dynamically

Value Object Examples

- U.S. ZIP Code
  - Not a number: 94115, 02134
- Birthdate
- Vehicle Identification Number (VIN)
- Entity Identifiers
  - UserId
  - CustomerId
  - OrderId

Money is Still a Primitive!



unconstrained, domain-free number

-2,147,483,648 2,147,483,647

`int`

1 100

`Wager`

constrained type has domain meaning

-999,999,999-01-01 *unconstrained date* +999,999,999-12-31

LocalDate

constrained type

2020-11-20 2020-12-18 2021-01-15 2021-02-19 2021-03-19 2021-04-16

ExpirationDate

Stateful Type

Usage: Holds Mutable State

Stateful Types

Playing Card Deck

Blackjack Hand

Order Quantity

Shipping Container

Refactor to Stateful Type

List<Card> ➡️ Deck, and Hand

Another Code Smell is Revealed

Feature Envy

Feature Envy

Game

```
private boolean beats(Hand hand, Hand otherHand) {
    return hand.value() > otherHand.value();
}

private boolean pushes(Hand hand, Hand otherHand) {
    return hand.value() == otherHand.value();
}

private boolean hasBlackjack(Hand hand) {
    return hand.value() == 21 && hand.getCards().size() == 2;
}

private boolean isBusted(Hand hand) {
    return hand.value() > 21;
}
```

Behavior implemented against data owned by another object

Fix Feature Envy: *Cohere Method*

Query methods
hide details,
expose logic
(decision-making)

Hand

```
boolean beats(Hand otherHand) {
    return value() > otherHand.value();
}

boolean pushes(Hand otherHand) {
    return value() == otherHand.value();
}

boolean hasBlackjack() {
    return value() == 21 && getCards().size() == 2;
}

boolean isBusted() {
    return value() > 21;
}
```

Fix Scalar Primitive Obsession

1. Fix Feature Envy: gather getter/setter usages
2. Remove getter & setter via Inline Method
3. Create new class with same field [Extract Delegate]
    1. Create getter/setter for field in new class
    2. Change old scalar to use instance of new type using search/replace
4. Extract and move Feature Envy methods
5. Remove getter & setter via Inline Method

Fix Collection Primitive Obsession

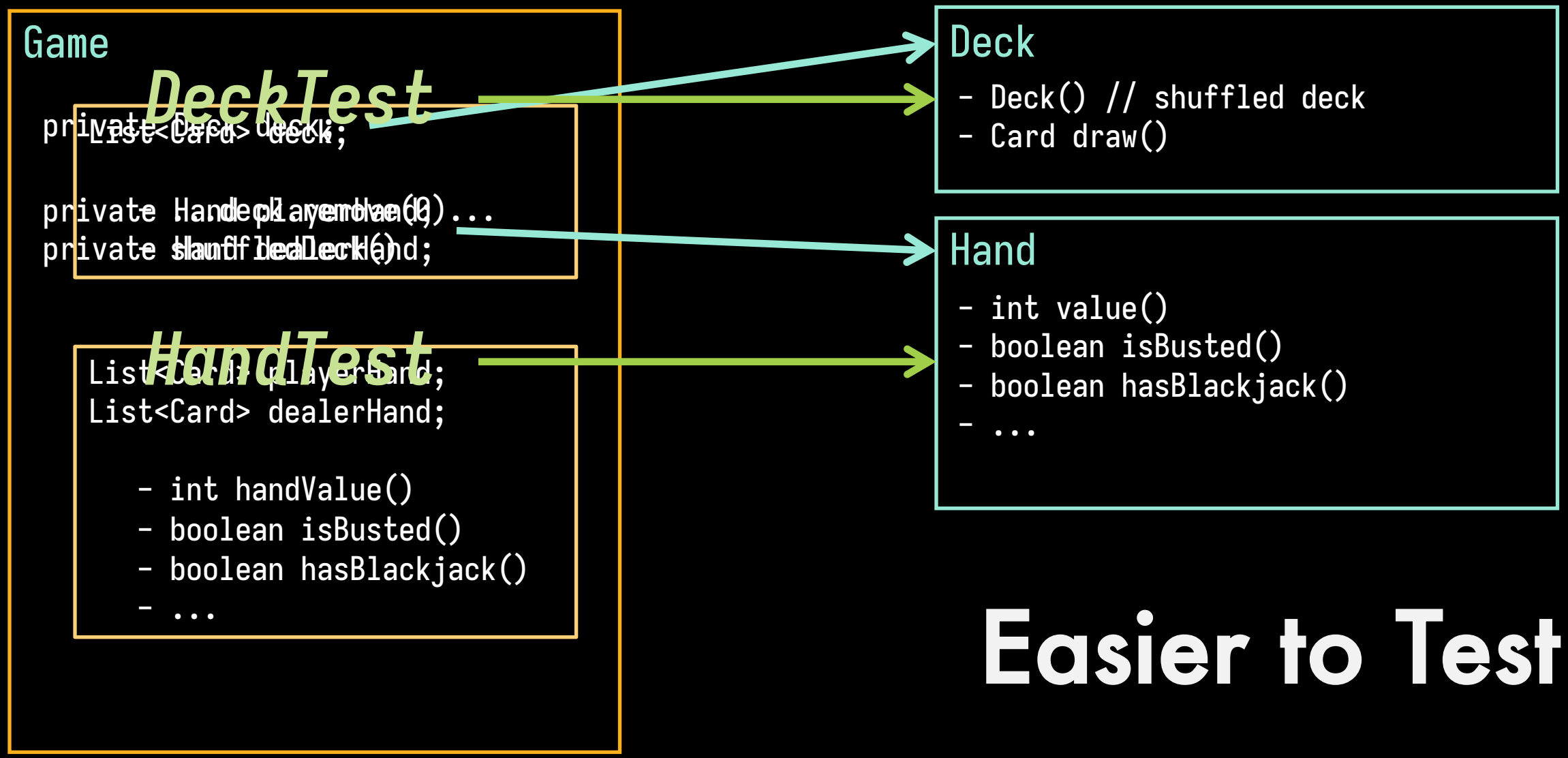
1. New class with getter for collection via Extract Delegate
2. Find usages of getter
3. Extract and Move Methods to new class
   - Use Introduce Parameter as needed
4. Continue until no more external use of getter
5. Inline getter to remove it
6. Clean up encapsulation, method names, etc.

Can also:
1. Manually move collection field to new class & create getter
2. Search/Replace use of field with a getter

Further Constrain Data Exposure

- Look for Feature Envy
  - Usages of **Query** methods that make decisions or calculations
- Look for public methods that
  - Can now be private
  - Return unconstrained types

More, Smaller Types



Easier to Test

Ted M. Young
Java Trainer, Coach, & Live Coder

Get in touch: ted@tedmyoung.com
About me: https://TED.Dev/about

Want your code to be easier to understand and test?

Fix Primitive Obsession with More, Smaller Types

Ted M. Young
Java Trainer, Coach, & Live Coder

Get in touch: ted@tedmyoung.com
Twitter: @JitterTed
Twitch: https://JitterTed.Stream
YouTube: https://JitterTed.TV
About me: https://TED.Dev/about

Thank You...

Source Code? Slides? Sign up at: https://mycmt.dev/