3-DAY BATTLE PLAN — WISE PAIR PROGRAMMING

No more reading. Only DOING.

DAY 0: TODAY (2-3 hours)

Hour 1: HackerRank Setup + Swift Reality Check

Step 1: Go to HackerRank RIGHT NOW

1. Open https://www.hackerrank.com
2. Sign up / Log in
3. Click "Prepare" → "Algorithms" → pick any problem
4. Select **Swift** from the language dropdown
5. Write this and RUN it:

```swift
```

```swift
import Foundation

// Test 1: Can you write a class?
class Cache<Key: Hashable, Value> {
    private var storage: [Key: Value] = [:]
    func get(_ key: Key) -> Value? { storage[key] }
    func set(_ key: Key, _ value: Value) { storage[key] = value }
}

// Test 2: Can you use Decimal?
let amount: Decimal = Decimal(string: "100.50")!
let fee = amount * Decimal(string: "0.005")!
print("Fee: \(fee)")  // 0.5025

// Test 3: Can you define a protocol?
protocol Convertible {
    func convert(amount: Decimal) -> Decimal
}

// Test 4: Can you use enums with associated values?
enum Result {
    case success(Decimal)
    case failure(String)
}

print("Swift works on HackerRank ✅ ")
```

THINGS THAT WORK DIFFERENTLY ON HACKERRANK:

- NO autocomplete — you must remember method names
- NO Xcode fix-it suggestions
- Read input with `readLine()!` (for algorithm problems)
- `import Foundation` is needed for Decimal, Date, etc.
- Compilation errors show at bottom — scroll down to see them
- You CAN'T run individual tests — it runs all at once

Swift syntax you MUST type from memory (no autocomplete):

```
Dictionary:   var dict: [String: Int] = [:]
              dict[key, default: 0] += 1
              dict.keys, dict.values

Array:        array.append(x)
              array.sorted { $0 < $1 }
```

```
            array.filter { $0 > 5 }
            array.map { $0 * 2 }
            array.reduce(0, +)
            array.first { $0 > 5 }
            array.enumerated()
            array.removeAll { condition }

String:     let chars = Array(str)
            String(chars[0...3])
            str.count (NOT str.length)
            str.contains("x")

Optional:   guard let x = optional else { return }
            let y = optional ?? defaultValue

Decimal:    Decimal(string: "0.92")!
            NOT Decimal(0.92) ← converts imprecise Double

Date:       Date()
            date.addingTimeInterval(30)
            date.timeIntervalSince(otherDate)

Print:      print("value: \(variable)")
```

Hour 2: Solve Circuit Breaker from SCRATCH on HackerRank

Set a 40-minute timer. Open a blank HackerRank editor. Swift. GO.

Don't look at the solution file. Try from memory. Talk out loud.

The problem: "Service B and C can fail. If either fails 3 times in 10 minutes, stop calling it for 5 minutes. Then try again."

Checkpoints:

- [ ] 0-5 min: Define `CircuitState` enum (closed, open, halfOpen)
- [ ] 5-10 min: Define `CircuitBreaker` class with properties
- [ ] 10-20 min: Implement `canExecute()`, `recordSuccess()`, `recordFailure()`
- [ ] 20-30 min: Sliding window for failures (remove old timestamps)
- [ ] 30-35 min: WebClient with per-service breakers
- [ ] 35-40 min: Write at least 1 test or discuss testing approach

If you get stuck: Open `wise_round2_manager_prep.md` and review, then close it and try again.

After finishing, ask yourself:

- Did I use a protocol for time? (Clock protocol for testability)
- Did I explain WHY for each decision?
- Did I mention Decimal if any financial values were involved?

Hour 3: Solve Cache with TTL from SCRATCH on HackerRank

Another 40-minute timer. Blank editor. Swift. GO.

The problem: "Build a cache with max size (LRU eviction) and TTL."

Checkpoints:

- [ ] 0-5 min: Define `Node` class with key, value, prev, next, createdAt
- [ ] 5-15 min: Basic get/set with doubly linked list + dictionary
- [ ] 15-25 min: LRU eviction when over capacity
- [ ] 25-35 min: TTL check on get — return nil if expired
- [ ] 35-40 min: Discuss thread safety (actor approach)

If you can't finish in 40 min: That's OK. The interview is collaborative — they'll guide you. But you need to get through the basic structure smoothly.

DAY 1: TOMORROW (3-4 hours)

Morning (2 hours): Remaining 3 problems on HackerRank

Problem 3: Currency Converter (40 min)

Start simple → they add fees → they add rate expiry

Checkpoints:

- [ ] Protocol `RateProvider`
- [ ] `CurrencyConverter` with Decimal (not Double!)
- [ ] Typed errors: `invalidAmount`, `rateNotAvailable`, `rateExpired`
- [ ] `FeeCalculator` as separate component
- [ ] Rate with TTL (30 seconds)

Problem 4: HTTP Client with Retry (40 min)

Basic fetch → add exponential backoff → 4xx vs 5xx

Checkpoints:

- [ ] Protocol `NetworkSession`
- [ ] Generic `fetch<T: Decodable>`

- [ ] Retry loop with configurable attempts
- [ ] `isRetryable()` — 5xx yes, 4xx no
- [ ] Exponential delay: `initialDelay * pow(multiplier, attempt)`

Problem 5: Merge Intervals (30 min)

Merge → Intersection → discuss Wise context

Checkpoints:

- [ ] Sort by start: `intervals.sorted { $0[0] < $1[0] }`
- [ ] Single-pass merge: compare current[0] with last[1]
- [ ] Two-pointer intersection
- [ ] State complexity out loud

Evening (1-2 hours): FULL MOCK SIMULATION

This is the most important practice session.

1. Set a 55-minute timer
2. Open HackerRank with blank Swift editor
3. Imagine an interviewer says: "We have services that can fail. How would you protect against cascading failures?"
4. DO THIS:
   - Spend 3-5 min asking clarifying questions (out loud to yourself)
   - Spend 5 min explaining your approach before coding
   - Code the solution, talking through every decision
   - When you finish the basic version, say "I'd add these improvements..."
   - Write or describe 2-3 tests
5. Record yourself on your phone (audio only is fine)
6. Listen back. Are you:
   - [ ] Explaining WHY at each step?
   - [ ] Mentioning protocols/testability?
   - [ ] Using Decimal for money?
   - [ ] Discussing edge cases?
   - [ ] Sounding collaborative ("What do you think?" "Should I handle this case?")

DAY 2: DAY BEFORE INTERVIEW (2 hours)

Morning (1 hour): Speed runs

Solve each of these in UNDER 15 MINUTES on HackerRank. No peeking:

Problem	Target Time	Key Thing to Not Forget
Circuit Breaker (basic)	15 min	Sliding window, per-service
LRU Cache (basic)	15 min	Doubly linked list, sentinel nodes
Currency Converter (basic)	10 min	Decimal, Protocol, typed errors
Two Sum	5 min	HashMap, one-pass

If any takes more than the target time, that's your weak spot — drill it once more.

Afternoon (1 hour): Review + Mental Prep

Re-read ONLY these sections:

1. `wise_round2_manager_prep.md` → "Manager-Level Behavioral Signals" table
2. `wise_round2_manager_prep.md` → "The 30-Second Rule" section
3. `wise_star_stories.md` → Story 1 (BFX) 30-second version only
4. `wise_star_stories.md` → Story 2 (RUNE) 30-second version only

Memorize these 5 phrases (say them out loud 3 times):

1. **Starting the interview:**
   > "Before I start coding, let me make sure I understand the requirements. Can I ask a few clarifying questions?"
2. **Choosing Decimal:**
   > "I'm using Decimal instead of Double because floating-point has precision issues that are unacceptable for financial calculations."
3. **Creating a protocol:**
   > "I'm extracting a protocol here so we can inject a mock for testing — and so the implementation can be swapped without changing callers."
4. **After finishing basic solution:**
   > "This works for the core case. If I had more time, I'd add thread safety with an actor, comprehensive error handling, and monitoring for production."
5. **When they ask about testing:**
   > "I'd inject a mock for the external dependency, test the happy path, test error cases, and test edge cases like empty input, zero amounts, and expired data."

Night before:

- [ ] Charge laptop, test internet connection
- [ ] Set up a quiet room with good lighting
- [ ] Have water nearby
- [ ] Sleep properly — a rested brain > 2 more hours of cramming

---

INTERVIEW DAY: 30 MIN BEFORE

- [ ] Open HackerRank, select Swift, write `print("ready")` — verify it works
- [ ] Close all other tabs/apps
- [ ] Put phone on silent
- [ ] Take 3 deep breaths
- [ ] Remember: they said "put your head together" — this is collaborative, not adversarial

---

DURING THE INTERVIEW: CHEAT SHEET

First 5 minutes — Introductions

> "Hi! I'm Keerthi, Lead iOS Engineer at PayPal Chennai. I've been working on foreign exchange systems and payment features for 8+ years. Excited to pair with you today."

When they describe the problem — DON'T CODE YET

Ask:

- "What are the expected inputs and outputs?"
- "Are there any edge cases you'd like me to handle?"
- "Should I focus on correctness first and optimize later?"
- "Any preference on approach — protocol-oriented, class-based?"

While coding — TALK CONSTANTLY

- "I'm starting with a protocol because..."
- "I'm using Decimal here because..."
- "This guard handles the edge case where..."
- "I could also do X, but Y is better because..."

When they suggest something — EMBRACE IT

- "That's a great point, let me incorporate that."
- "I hadn't considered that — here's how I'd adjust..."
- Never argue. They're testing collaboration.

When you're stuck — DON'T FREEZE

- "Let me think about this for a moment..."
- "I'm considering two approaches: A and B. A is simpler but B handles... What do you think?"
- Ask them: "Would you suggest I start with the simpler approach and iterate?"

After coding — VOLUNTEER IMPROVEMENTS

- "For production, I'd add: logging, monitoring, feature flags"
- "For testing, I'd inject mocks for..."
- "The thread safety concern is... I'd solve it with an actor"

WHAT NOT TO DO

❌ Don't	✅ Do Instead
Jump straight to coding	Ask 2-3 clarifying questions first
Code silently	Narrate every decision
Use Double for money	Always Decimal with string init
Force unwrap (!)	guard let or ??
Write everything in one function	Separate concerns: protocol, types, logic
Say "I don't know" and stop	Say "I'm not sure, but my instinct is... what do you think?"
Ignore their suggestions	"Great idea, let me add that"
Forget testing	"Here's how I'd test this..." (even if you don't write tests)
Try to be perfect	Working code + clean structure > perfect code unfinished