```
# In your main command class
def __init__(self, *args, **kwargs):
    super().__init__(*args, **kwargs)
    # Initialize with debug callback
    self.class_analyzer = ClassAnalysisService(debug_callback=self.stdout.write)
    self.run_analyzer = RunAnalysisService()
    self.jt_analysis_cache = {}

def calculate_horse_score(self, horse):
    """Calculate a comprehensive score for a horse with detailed debug"""
    self.stdout.write(f"\n{'='*60}")
    self.stdout.write(f"🏇 CALCULATING SCORE FOR {horse.horse_name} (#{horse.horse_no})")
    self.stdout.write(f"{'='*60}")

    # Run analysis
    run_analysis = self.run_analyzer.analyze_horse_runs(horse)
    self.stdout.write(f"📊 Run analysis - Form: {run_analysis.get('form_rating', 0):.2f}, Consistency:
{run_analysis.get('consistency', 0):.1f}%")

    # Class analysis with detailed debug
    class_suitability = self.class_analyzer.calculate_class_suitability(horse, horse.race)
    class_trend = self.class_analyzer.get_class_trend(horse)

    # Get detailed class history
    class_history = self.class_analyzer.analyze_horse_class_history(horse)
    if class_history['run_analyses']:
        run_details = [f"{a['class_group']}({a['run_score']:.1f})" for a in class_history['run_analyses']]
        self.stdout.write(f"📈 Class history: {run_details}")
        self.stdout.write(f"⭐ Best performance: {class_history['best_performance']['class_group'] if
class_history['best_performance'] else 'None'}")

    # Base score components
    merit_score = horse.horse_merit or 0
    form_score = 100 - (run_analysis.get('form_rating', 0) * 5)
    consistency_score = run_analysis.get('consistency', 50)

    # J-T score from cache
    jt_score = 50
    if horse.horse_no in self.jt_analysis_cache:
        jt_score = self.jt_analysis_cache[horse.horse_no]['score']
        self.stdout.write(f"👥 J-T Score: {jt_score}")

    # Final score calculation
    score = (
        (merit_score * 0.25) +
        (class_suitability * 0.25) +
        (form_score * 0.2) +
        (consistency_score * 0.15) +
        (jt_score * 0.15)
```

```
)

self.stdout.write(f"\n🧮 FINAL SCORE BREAKDOWN:")
self.stdout.write(f"   Merit: {merit_score} × 0.25 = {merit_score * 0.25:.2f}")
self.stdout.write(f"   Class: {class_suitability} × 0.25 = {class_suitability * 0.25:.2f}")
self.stdout.write(f"   Form: {form_score:.1f} × 0.20 = {form_score * 0.2:.2f}")
self.stdout.write(f"   Consistency: {consistency_score} × 0.15 = {consistency_score * 0.15:.2f}")
self.stdout.write(f"   J-T: {jt_score} × 0.15 = {jt_score * 0.15:.2f}")
self.stdout.write(f"   TOTAL: {score:.2f}")
self.stdout.write(f"{'='*60}")

return {
    'horse': horse,
    'score': round(score, 2),
    'merit_score': merit_score,
    'class_score': class_suitability,
    'form_score': round(form_score, 2),
    'consistency_score': round(consistency_score, 2),
    'jt_score': jt_score,
    'class_trend': class_trend
}
```