Take-Home: ERC-721 with Harberger Taxes
==========

Brief

Design and implement an ERC-721 NFT that uses Harberger taxes. We want to see your protocol thinking, edge-case handling, and implementation quality—not just a compiling contract.

Constraints

- Build the contract and tests yourself; don't use AI to write code.
- Use a recent, stable Solidity version.
- Tooling: Foundry only (forge build/test/anvil).

Core Requirements

---
1) ERC-721 basics
  - Standard mint/transfer/ownerOf/balanceOf behavior.
  - You may extend OpenZeppelin or write your own; be explicit about which parts are reused.

2) Self-assessed pricing
  - Each token stores an on-chain price set by its owner and visible to anyone.
  - The token must always be purchasable at that price. Owners can update it; document any rules you impose (minimums, rate limits, requiring taxes be paid first, etc.).

3) Buying at the declared price
  - Anyone can buy a token by paying its current price (refund excess).
  - Handle fund flows (seller vs. protocol/treasury) and unpaid taxes at sale time.
  - Ensure there are no hidden blockers to purchasing at the posted price.

4) Harberger tax accrual
  - Define a tax rate (global or per-token).
  - Accrue tax based on price and elapsed time; settle on interactions (buy, update price, pay tax, etc.).
  - Track whatever state you need (e.g., price, last-settled timestamp).

- Specify how tax is paid and where it goes (e.g., treasury address). Document your formula and choices.

5) Handling non-payment
- Propose and implement a clear default/foreclosure path (examples: repossession/burn after threshold, prepaid deposit that must stay funded, blocking operations until taxes are settled).
- Justify your approach and edge-case handling.

Non-Functional Expectations

- Security: consider reentrancy, overflow/underflow (by Solidity version), rounding, unsafe external calls, and extreme inputs (very high prices/tax rates).
- Gas/UX: keep frequent calls reasonable; consider long time gaps in tax math; make integration straightforward.

Deliverables
---
1) Smart contracts: ERC-721 + Harberger logic.
2) Tests: cover minting, price updates, buying, accrual over time, paying taxes, your default/foreclosure path, and edge cases (price changes, long intervals, underpayment, etc.).
3) Short design note (1–2 pages): describe your tax model (formula, settlement timing, destination of funds), your default/foreclosure rules, and any trade-offs or simplifications. Note any intentional deviations from this spec.
4) README: how to install deps, compile, and run tests with Foundry (`forge install`, `forge build`, `forge test`).

What We're Evaluating

- Clarity of design.
- Correctness and robustness of the contract.
- Quality and coverage of tests.
- Practical choices around gas and UX.

Submission

Share the repo with contracts, tests, design note, and README, with the githubs @jnoorchashm37 and @will_Smith11.