[](https://ida.interchain.io/)

[Interchain Developer Academy](https://ida.interchain.io/)/[Interchain Developer Academy](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)



Search

[Interchain Developer Academy](https://ida.interchain.io/)[Interchain Developer Academy](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

Search



Filters

Interchain Developer Academy

[](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Week 0 - Getting Started](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Getting Started](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Blockchain 101](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Blockchain History](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Public and Managed Blockchains](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Consensus in Distributed Networks](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Cryptography](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Self-Assessment Quiz](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Go Introduction - First Steps](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Go Basics](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Go Interfaces](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Control Structures in Go](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Arrays and Slices in Go](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Standard Packages in Go](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Concurrency in Go](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Good-To-Know Dev Terms](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Docker Introduction](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Week 1 - Introduction to the Interchain](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Introduction to the Interchain](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Blockchain Technology and the Interchain](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[The Interchain Ecosystem](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Getting ATOM and Staking It](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[A Blockchain App Architecture](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Accounts](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Transactions](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Messages](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Modules](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Protobuf](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Multistore and Keepers](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[BaseApp](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Queries](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Events](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Context](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Testing](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Relaying with IBC](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Interchain Security](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Bridges](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Migrations](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Week 1 Quiz](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Week 2 - First Steps](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[First Steps](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Setup Your Work Environment](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Run a Node, API, and CLI](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Ignite CLI](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Exercise - Make a Checkers Blockchain](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Store Object](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Create Custom Messages](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Create and Save a Game Properly](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Add a Way to Make a Move](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Emit Game Information](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Record the Game Winner](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Week 2 Exercise](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Week 3 - Introduction to IBC and CosmJS](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Introduction to IBC and CosmJS](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[What is IBC?](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[IBC/TAO - Connections (OPTIONAL)](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[IBC/TAO - Channels (OPTIONAL)](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[IBC/TAO - Clients (OPTIONAL)](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[IBC Token Transfer](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Interchain Accounts (OPTIONAL)](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[IBC Middleware (OPTIONAL)](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Create a Custom IBC Middleware (OPTIONAL)](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Integrate IBC Middleware Into a Chain (OPTIONAL)](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[IBC Tooling](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[What is CosmJS?](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Your First CosmJS Actions](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Compose Complex Transactions](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Learn to Integrate Keplr](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Create Custom CosmJS Interfaces](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Week 4 - Ignite CLI and IBC Advanced](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Ignite CLI and IBC Advanced](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Keep an Up-To-Date Game Deadline](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Keep Track Of How Many Moves Have Been Played](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Put Your Games in Order](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Auto-Expiring Games](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Let Players Set a Wager](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Handle wager payments](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Integration tests](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Incentivize Players](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Help Find a Correct Move](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Play With Cross-Chain Tokens](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Understand IBC Denoms](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Go Relayer](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Hermes Relayer](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Week 5 - CosmJS Advanced](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[CosmJS Advanced](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Create Custom Objects](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Create Custom Messages](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Get an External GUI](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Integrate CosmJS and Keplr](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Backend Script for Game Indexing](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Week 6 - IBC Deep Dive](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[IBC Deep Dive](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[IBC Application Developer Introduction](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Make a Module IBC-Enabled](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Adding Packet and Acknowledgment Data](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Extend the Checkers Game With a Leaderboard](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Create a Leaderboard Chain](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Week 7 - From Code to MVP to Production and Migrations](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[From Code to MVP to Production and Migrations](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Run in Production](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Prepare the Software to Run](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Prepare a Validator and Keys](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Prepare Where the Node Starts](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Prepare and Connect to Other Nodes](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Configure, Run, and Set Up a Service](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Prepare and Do Migrations](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Simulate Production in Docker](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Tally Player Info After Production](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Add a Leaderboard as a Module](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Migrate the Leaderboard Module After Production](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Simulate a Migration in Docker](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Final Exam](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[What's Next?](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

[Continue Your Interchain Journey](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html)

Docs Version Switcher

On this page

[High level considerations](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#high-level-considerations)

[What you will do](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#what-you-will-do)

[New v2 module](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#new-v2-module)

[New v2 information](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#new-v2-information)

[Transient object](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#transient-object)

[Leaderboard helpers](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#leaderboard-helpers)

[Candidate Lifecycle](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#candidate-lifecycle)

[Prepare transient store](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#prepare-transient-store)

[Prepare candidate store keys](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#prepare-candidate-store-keys)

[Use the candidate store](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#use-the-candidate-store)

[Leaderboard handling](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#leaderboard-handling)

[Hook infrastructure for candidates](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#hook-infrastructure-for-candidates)

[On the checkers module](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#on-the-checkers-module)

[On the leaderboard module](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#on-the-leaderboard-module)

[On app.go](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#on-app-go)

[Unit tests](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#unit-tests)

[Candidate unit tests](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#candidate-unit-tests)

[Leaderboard helper unit tests](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#leaderboard-helper-unit-tests)

[Candidate lifecycle unit tests](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#candidate-lifecycle-unit-tests)

[Leaderboard handling unit tests](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#leaderboard-handling-unit-tests)

[Hook unit tests on leaderboard](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#hook-unit-tests-on-leaderboard)

[Hook unit tests on checkers](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#hook-unit-tests-on-checkers)

[Integration tests](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#integration-tests)

[Interact via the CLI](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#interact-via-the-cli)

[#Copy link](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#add-a-leaderboard-module) **Add a Leaderboard Module**



Make sure you have all you need before proceeding:

* You understand the concepts of [Protobuf](https://ida.interchain.io/academy/2-cosmos-concepts/6-protobuf.html), [modules](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html), and [migrations](https://ida.interchain.io/academy/2-cosmos-concepts/16-migrations.html).
* Go is installed.
* You have the checkers blockchain codebase up to the *Tally Player Info After Production*. If not, follow the [previous steps](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/2-migration-info.html) or check out the [relevant version (opens new window)↗](https://github.com/cosmos/b9-checkers-academy-draft/tree/player-info-migration).



In this section, you will:

* Add a new module.
* Add a leaderboard storage type.
* Add hooks for loose coupling of modules.
* Use the transient store.

In the previous section you added a player info structure that tallies wins and losses per player. On its own, this information could be collected outside of the blockchain via a [dedicated server](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html).

It was in fact done on-chain so as to make this new step more relevant. If you want an on-chain leaderboard that is provably correct, then you need its information to come from the chain too. As a result of this choice, you have the necessary information on-chain in the form of PlayerInfo. You now need to organize it into a leaderboard.

[#Copy link](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#high-level-considerations) High level considerations

Your blockchain is now at *v1.1*. In this section, you will introduce *v2* of your blockchain with leaderboard support. A good leaderboard fulfills these conditions:

* Any player who has **ever** played should have a tally of games won, lost, and forfeited. You already have that.
* The leaderboard should list the players with the most wins up to a pre-determined number of players. For example, the leaderboard might only include the top 100 scores.
* To avoid squatting and increase engagement, when scores are equal in value the most recent score takes precedence over an *older* one: the player with the **more recent score** is listed higher on the leaderboard.



When you introduce the leaderboard in production, you also have to consider migration. This concern is covered in the [next section](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/4-migration-leaderboard.html).

The leaderboard is not strictly the concern of the game of checkers. It is a side concern. The concept of a leaderboard is also very generic, you could easily imagine it being used for other types of game. Therefore, it makes sense to introduce it as a **separate module** next to the checkers module.

The checkers and leaderboard modules will exchange information. More specifically, the leaderboard needs to know when a player's total wins change, as this may warrant entering the leaderboard. If you have the checkers module call the leaderboard module (just as it calls the bank when handling wagers), the checkers module needs to know the details of the leaderboard module. It is best to avoid such tight coupling.

Fortunately, you can reuse a ***hooks* pattern** already used in the Cosmos SDK. With this future addition, the leaderboard module adds a listener to the hook interface of the checkers module. With this the checkers module informs any listeners, whether there are none, one, or many.

The leaderboard module will work by listening to results from the checkers module. It will not have messages of it own.

Thinking about early performance optimization, you have to decide what operations the module does when it receives one candidate from the checkers module. The first idea is to:

1. Read the leaderboard from storage, which includes all 100 members.
2. Conditionally add the candidate to the leaderboard.
3. If added, sort and clip the list.
4. Put the leaderboard back in storage.

These are a lot of expensive operations for a single candidate.

Fortunately, there is a better way. The leaderboard needs to be computed and saved when the block is prepared, but it does not need to be up to date after each (checkers) transaction. You can imagine keeping the leaderboard (or something approximating it) in memory for the whole length of the block.

In the section about [expiring games](https://ida.interchain.io/hands-on-exercise/2-ignite-cli-adv/4-game-forfeit.html), you learned about EndBlock. There is also a BeginBlock callback. It is conceivable to prepare the leaderboard in BeginBlock and **keep it in the context or a memory or transient storage**. Then it would be recalled with each candidate, and finally (in EndBlock, and only there) it would be sorted and clipped before being saved in storage proper.

Better still, though, you do not need to *prepare* the leaderboard in BeginBlock. You can just keep candidates in the transient storage as they come. Then only in EndBlock is the leaderboard loaded, updated, and saved.

[#Copy link](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#what-you-will-do) What you will do

Several things need to be addressed to build your v2 blockchain:

1. Add the leaderboard module.
2. Define your new data types.
3. Add helper functions to encapsulate clearly defined actions, like leaderboard sorting.
4. Prepare keys to store candidates in a transient store.
5. Adjust the existing code to make use of and update the new data types.
6. Add the hooks pattern elements.
7. Handle the leaderboard properly.
8. Configure the app for it.

[#Copy link](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#new-v2-module) New v2 module

As discussed, you will introduce a new leaderboard module. This is conveniently done with Ignite CLI.

Ignite also offers the possibilty to add new Params to the module. These are module-wide parameters:

1. Whose original value is defined in the genesis
2. That can be modified via governance proposal

It could be interesting to have the length of the leaderboard be defined like that.

**Local**

**Docker**



Copy

$ ignite scaffold module leaderboard \

--params length:uint

Copy

$ docker run --rm -it \

-v $(pwd):/checkers \

-w /checkers \

checkers\_i \

ignite scaffold module leaderboard \

--params length:uint

With that, Ignite has created a new [x/leaderboard (opens new window)↗](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-module/x/leaderboard) folder next to x/checkers. It has also put a length field inside Params:



Copy

message Params {

...

uint64 length = 1 [(gogoproto.moretags) = "yaml:\"length\""];

}

proto /

leaderboard /

params.proto

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-module/proto/leaderboard/params.proto" \l "L12" \t "_blank)

The genesis defines a [starting value (opens new window)↗](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-module/x/leaderboard/types/genesis.go#L14) of 0 for this length. You ought to change it now to something adequate:



Copy

var (

KeyLength = []byte("Length")

- // TODO: Determine the default value

- DefaultLength uint64 = 0

+ )

+

+ const (

+ DefaultLength uint64 = 100

)

x /

leaderboard /

types /

params.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-objects/x/leaderboard/types/params.go" \l "L14" \t "_blank)

You also make it const as this is the case and becomes helpful later on.

[#Copy link](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#new-v2-information) New v2 information

It is time to take a closer look at the new data structures being introduced with the new module.



If you feel unsure about creating new data structures with Ignite CLI, look at the [previous sections](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/4-create-message.html) of the exercise again.

To give the new v2 information a data structure, you need the following:

1. Add a structure for **the leaderboard**: you want a single stored leaderboard for the whole module. Let Ignite CLI help you implement a structure:

**Local**

**Docker**



Copy

$ ignite scaffold single leaderboard winners \

--module leaderboard --no-message

Copy

$ docker run --rm -it \

-v $(pwd):/checkers \

-w /checkers \

checkers\_i \

ignite scaffold single leaderboard winners \

--module leaderboard --no-message

1. This creates a Protobuf file with string winners. This is not very useful. So you declare by hand another Protobuf message in leaderboard.proto for use as a leaderboard rung:



Copy

message Leaderboard {

string winners = 1;

}

+ message Winner {

+ string address = 1;

+ uint64 wonCount = 2;

+ uint64 addedAt = 3;

+ }

proto /

leaderboard /

leaderboard.proto

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-object/proto/leaderboard/leaderboard.proto" \l "L11-L15" \t "_blank)



Take note of the key features:

* + address indicates the player. This will be the same address as the one that comes in PlayerInfo.index.
  + wonCount determines the ranking on the leaderboard - the higher the count, the closer to the 0 index in the array. This should exactly match the value found in the corresponding player stats. This duplication of data is a lesser evil, because if wonCount was missing you would have to access the player stats to sort the leaderboard.
  + addedAt is a timestamp that indicates when the player's wonCount was last updated and determines the ranking when there is a tie in wonCount - the more recent, the closer to the 0 index in the array.

1. You make the Leaderboard message use message Winner as an array. Add that each element in the map is not nullable. This will compile each WinningPlayer to a Go object instead of a pointer:



Copy

+ import "gogoproto/gogo.proto";

message Leaderboard {

- string winners = 1;

+ repeated WinningPlayer winners = 1 [(gogoproto.nullable) = false];

}

proto /

leaderboard /

leaderboard.proto

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-objects/proto/leaderboard/leaderboard.proto" \l "L5-L8" \t "_blank)

1. The v2 genesis was also updated with the leaderboard. Tell it that the leaderboard should always be there (even if empty):



Copy

message GenesisState {

...

- Leaderboard leaderboard = 2;

+ Leaderboard leaderboard = 2 [(gogoproto.nullable) = false];

}

proto /

leaderboard /

genesis.proto

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-objects/proto/leaderboard/genesis.proto" \l "L14" \t "_blank)

At this point, you should run ignite generate proto-go so that the corresponding Go objects are re-created.

1. Remember to make sure the initial value stored for the leaderboard is not nil but instead is an empty list. In genesis.go adjust:



Copy

func DefaultGenesis() \*GenesisState {

return &GenesisState{

- Leaderboard: nil,

+ Leaderboard: Leaderboard{

+ Winners: []Winner{},

+ },

...

}

}

x /

leaderboard /

types /

genesis.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-objects/x/leaderboard/types/genesis.go" \l "L13-L15" \t "_blank)

This function returns a default genesis. This step is important if you start fresh. In your case, you do not begin with an "empty" genesis but with one resulting from the upcoming genesis migration in this exercise.

In particular, add a test on the initial genesis:



Copy

func TestDefaultGenesisState\_ExpectedInitial(t \*testing.T) {

require.EqualValues(t,

&types.GenesisState{

Leaderboard: types.Leaderboard{

Winners: []types.Winner{},

},

Params: types.Params{

Length: 100,

},

},

types.DefaultGenesis())

}

x /

leaderboard /

types /

genesis\_test.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-objects/x/leaderboard/types/genesis_test.go" \l "L61-L72" \t "_blank)

Fix the compilation error in the same file:



Copy

- Leaderboard: &types.Leaderboard{

- Winners: "49",

+ Leaderboard: types.Leaderboard{

+ Winners: []types.Winner{},

},

x /

leaderboard /

types /

genesis\_test.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-objects/x/leaderboard/types/genesis_test.go" \l "L25-L27" \t "_blank)

And add a test case that catches a duplicated winner address:



Copy

{

desc: "duplicated winnerPlayer",

genState: &types.GenesisState{

Leaderboard: types.Leaderboard{

Winners: []types.Winner{

{

Address: "cosmos123",

},

{

Address: "cosmos123",

},

},

},

},

valid: false,

},

x /

leaderboard /

types /

genesis\_test.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-objects/x/leaderboard/types/genesis_test.go" \l "L32-L47" \t "_blank)

1. Also adjust other compilation errors:

On genesis.go:



Copy

- // Set if defined

- if genState.Leaderboard != nil {

- k.SetLeaderboard(ctx, \*genState.Leaderboard)

- }

+ k.SetLeaderboard(ctx, genState.Leaderboard)

x /

leaderboard /

genesis.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-objects/x/leaderboard/genesis.go" \l "L12" \t "_blank)

And:



Copy

if found {

- genesis.Leaderboard = &leaderboard

+ genesis.Leaderboard = leaderboard

}

x /

leaderboard /

genesis.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-objects/x/leaderboard/genesis.go" \l "L25" \t "_blank)

On genesis\_test.go:



Copy

...

- Leaderboard: &types.Leaderboard{

- Winners: "94",

+ Leaderboard: types.Leaderboard{

+ Winners: []types.Winner{

+ {

+ Address: "cosmos123",

+ },

+ {

+ Address: "cosmos456",

+ },

+ },

},

...

x /

leaderboard /

genesis\_test.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-objects/x/leaderboard/genesis_test.go" \l "L36-L45" \t "_blank)

On client/cli/query\_leaderboard\_test.go:



Copy

- leaderboard := &types.Leaderboard{}

+ leaderboard := types.Leaderboard{}

nullify.Fill(&leaderboard)

state.Leaderboard = leaderboard

buf, err := cfg.Codec.MarshalJSON(&state)

require.NoError(t, err)

cfg.GenesisState[types.ModuleName] = buf

- return network.New(t, cfg), \*state.Leaderboard

+ return network.New(t, cfg), state.Leaderboard

x /

leaderboard /

... /

cli /

query\_leaderboard\_test.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-objects/x/leaderboard/client/cli/query_leaderboard_test.go" \l "L24-L30" \t "_blank)

1. Now that the leaderboard will always be in the store, you may as well change the GetLeaderboard function so that it panics instead of returning an error when it cannot find it:



Copy

- func (k Keeper) GetLeaderboard(ctx sdk.Context) (val types.Leaderboard, found bool) {

+ func (k Keeper) GetLeaderboard(ctx sdk.Context) (val types.Leaderboard) {

...

if b == nil {

- return val, false

+ panic("Leaderboard not found")

}

k.cdc.MustUnmarshal(b, &val)

- return val, true

+ return val

}

x /

leaderboard /

keeper /

leaderboard.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-objects/x/leaderboard/keeper/leaderboard.go" \l "L17-L27" \t "_blank)

This requires further easy fixing of new compilation errors. A more complex fix is the one that checks what happens when the leaderboard is removed:



Copy

func TestLeaderboardRemove(t \*testing.T) {

keeper, ctx := keepertest.LeaderboardKeeper(t)

createTestLeaderboard(keeper, ctx)

keeper.RemoveLeaderboard(ctx)

- \_, found := keeper.GetLeaderboard(ctx)

- require.False(t, found)

+ defer func() {

+ r := recover()

+ require.NotNil(t, r, "The code did not panic")

+ require.Equal(t, r, "Leaderboard not found")

+ }()

+ keeper.GetLeaderboard(ctx)

}

x /

leaderboard /

keeper /

leaderboard\_test.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-objects/x/leaderboard/keeper/leaderboard_test.go" \l "L31-L41" \t "_blank)

1. The test case you added will fail unless you update the Validate() method of the genesis to not allow duplicate player addresses. This is inspired by types/genesis.go, and is best kept in a separate and new types/leaderboard.go:



Copy

func (leaderboard Leaderboard) Validate() error {

// Check for duplicated player address in winners

winnerInfoIndexMap := make(map[string]struct{})

for index, elem := range leaderboard.Winners {

if \_, ok := winnerInfoIndexMap[elem.Address]; ok {

return fmt.Errorf("duplicated address %s at index %d", elem.Address, index)

}

winnerInfoIndexMap[elem.Address] = struct{}{}

}

return nil

}

x /

leaderboard /

types /

leaderboard.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-objects/x/leaderboard/types/leaderboard.go" \t "_blank)

After this, you can adjust the types/genesis.go file:



Copy

func (gs GenesisState) Validate() error {

+ // Validate Leaderboard

+ if err := gs.Leaderboard.Validate(); err != nil {

+ return err

+ }

// this line is used by starport scaffolding # genesis/types/validate

...

}

x /

checkers /

types /

genesis.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-objects/x/checkers/types/genesis.go" \l "L24-L27" \t "_blank)

You can confirm that the existing unit tests pass.

[#Copy link](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#transient-object) Transient object

You will use objects when storing candidates in a transient KVStore between BeginBlock and EndBlock. You want them to be small.

In leaderboard.proto, add:



Copy

message Winner {

...

}

+ message Candidate {

+ bytes address = 1;

+ uint64 wonCount = 2;

+ }

proto /

leaderboard /

leaderboard.proto

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-object/proto/leaderboard/leaderboard.proto" \l "L17-L20" \t "_blank)

Where bytes address is the player's undecoded address.

Remember that sdk.AccAddress's underlying type is byte[].

After another round of Go compilation, you can add a helper function to get a Candidate's address as a Bech32 string:



Copy

func (candidate Candidate) GetAccAddress() string {

return sdk.AccAddress(candidate.Address).String()

}

x /

leaderboard /

types /

leaderboard.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-objects/x/leaderboard/types/leaderboard.go" \t "_blank)

Where sdk.AccAddress(candidate.Address) is casting the byte[] into sdk.AccAddress.

Also add a function to convert it into a leaderboard rung at a given time:



Copy

func (candidate Candidate) GetWinnerAtTime(now time.Time) Winner {

return Winner{

Address: candidate.GetAccAddress(),

WonCount: candidate.WonCount,

AddedAt: uint64(now.Unix()),

}

}

x /

leaderboard /

types /

leaderboard.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-objects/x/leaderboard/types/leaderboard.go" \t "_blank)

With the structure set up, it is time to add the code using these new elements in normal (non-migration) operations.

[#Copy link](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#leaderboard-helpers) Leaderboard helpers

Continue working on your v2 before tackling the migration. In both the migration and regular operations, the leaderboard helpers have to:

1. Add a number of new candidates to your array of winners.
2. Sort the array according to the rules.
3. Clip the array to the chosen length and save the result.

You can reuse your types/leaderboard.go to encapsulate all your leaderboard helpers:

1. Add functions to sort a slice of winners in place:



Copy

func SortWinners(winners []Winner) {

sort.SliceStable(winners[:], func(i, j int) bool {

if winners[i].WonCount > winners[j].WonCount {

return true

}

if winners[i].WonCount < winners[j].WonCount {

return false

}

return winners[i].AddedAt > winners[j].AddedAt

})

}

func (leaderboard Leaderboard) SortWinners() {

SortWinners(leaderboard.Winners)

}

x /

leaderboard /

types /

leaderboard.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-handling/x/leaderboard/types/leaderboard.go" \l "L36-L50" \t "_blank)

It tests in descending order, first for scores and then for the timestamps.



It is possible to write a one-liner inside this function, but at the expense of readability.

1. When it comes to adding or updating candidates to the array of winners, your goal is to make these operations as efficient as possible. To avoid having to find duplicate player addresses in an array, it is better to use a map. Add a function to convert an array of winners into a map:



Copy

func MapWinners(winners []Winner, length int) map[string]Winner {

mapped := make(map[string]Winner, length)

for \_, winner := range winners {

already, found := mapped[winner.Address]

if !found {

mapped[winner.Address] = winner

} else if already.WonCount < winner.WonCount {

mapped[winner.Address] = winner

}

}

return mapped

}

x /

leaderboard /

types /

leaderboard.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-handling/x/leaderboard/types/leaderboard.go" \l "L52-L63" \t "_blank)

1. The timestamp used when a winner is added to the leaderboard will be the block's time. In other words, it will be the same time for all candidates added in EndBlock. Prepare a function to do that:



Copy

func AddCandidatesAtNow(winners []Winner, now time.Time, candidates []Candidate) (updated []Winner) {

mapped := MapWinners(winners, len(winners)+len(candidates))

for \_, candidate := range candidates {

if candidate.WonCount < 1 {

continue

}

candidateWinner := candidate.GetWinnerAtTime(now)

already, found := mapped[candidateWinner.Address]

if !found {

mapped[candidateWinner.Address] = candidateWinner

} else if already.WonCount < candidateWinner.WonCount {

mapped[candidateWinner.Address] = candidateWinner

}

}

updated = make([]Winner, 0, len(mapped))

for \_, winner := range mapped {

updated = append(updated, winner)

}

SortWinners(updated)

return updated

}

x /

checkers /

types /

leaderboard.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-migration/x/checkers/types/leaderboard.go" \l "L112-L155" \t "_blank)

Note how, when creating the map, it initializes with a capacity equal to the sum of both winners and candidates' lengths. This is an approximative way of increasing memory performance.

[#Copy link](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#candidate-lifecycle) Candidate Lifecycle

You have prepared helper functions that will update a list of winners with a list of candidates. The candidates will come from the transient store – transient in the sense that it will be discarded after EndBlock. That is good for this usage, as you do not want to carry candidates from one block to the next.

Your leaderboard module does not have access to a transient store by default, so you will have to prepare that first.

Additionally, you want to reduce the number of marshalling / unmarshalling taking place repeatedly. It would not make sense to unmarshall a whole array of candidates every time you want to add a single candidate to the array. Instead, it makes sense to keep each candidate as a single entry in the store, and separately keep the information on how many n are being stored. Later, you can retrieve them with [k] where 0 <= k < n.

You will:

1. Prepare your leaderboard module with access to a transient store.
2. Define keys of the candidates transient store.
3. Add a function to prepare the candidates transient store in BeginBlock.
4. Add a function to add a single candidate to the store.
5. Add a function to retrieve all the candidates from the transient store.

[#Copy link](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#prepare-transient-store) Prepare transient store

By default, Ignite CLI does not prepare your module to have access to a transient store like it prepares it to have access to the proper store. The preparation works the same way as a normal store.

Add a transient store key in your keeper:



Copy

type (

Keeper struct {

...

memKey sdk.StoreKey

+ tKey sdk.StoreKey

paramstore paramtypes.Subspace

}

)

x /

leaderboard /

keeper /

keeper.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-handling/x/leaderboard/keeper/keeper.go" \l "L19" \t "_blank)

Update the constructor accordingly:



Copy

func NewKeeper(

...

memKey sdk.StoreKey,

+ tKey sdk.StoreKey,

ps paramtypes.Subspace,

) \*Keeper {

...

return &Keeper{

...

memKey: memKey,

+ tKey: tKey,

paramstore: ps,

}

}

x /

leaderboard /

keeper /

keeper.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-handling/x/leaderboard/keeper/keeper.go" \l "L24-L45" \t "_blank)

This key will be identified by a new string in app.go's list of transient store keys. Add such a distinct key:



Copy

MemStoreKey = "mem\_leaderboard"

+

+ // TStoreKey defines the transient store key

+ TStoreKey = "transient\_leaderboard"

x /

leaderboard /

types /

keys.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-handling/x/leaderboard/types/keys.go" \l "L19-L20" \t "_blank)

Adjust app.go so that it gives the keeper a valid key. Also take this opportunity to fix an Ignite bug on memKeys:



Copy

app.LeaderboardKeeper = \*leaderboardmodulekeeper.NewKeeper(

...

- keys[leaderboardmoduletypes.MemStoreKey],

+ memKeys[leaderboardmoduletypes.MemStoreKey],

+ tkeys[leaderboardmoduletypes.TStoreKey],

app.GetSubspace(leaderboardmoduletypes.ModuleName),

)

app /

app.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-handling/app/app.go" \l "L419-L420" \t "_blank)

Do not forget to ensure that there is indeed a store key at the string(s) you asked:



Copy

- tkeys := sdk.NewTransientStoreKeys(paramstypes.TStoreKey)

- memKeys := sdk.NewMemoryStoreKeys(capabilitytypes.MemStoreKey)

+ tkeys := sdk.NewTransientStoreKeys(paramstypes.TStoreKey, leaderboardmoduletypes.TStoreKey)

+ memKeys := sdk.NewMemoryStoreKeys(capabilitytypes.MemStoreKey, leaderboardmoduletypes.MemStoreKey)

app /

app.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-handling/app/app.go" \l "L303-L304" \t "_blank)

[#Copy link](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#prepare-candidate-store-keys) Prepare candidate store keys

Your keeper has access to a transient store. Define the keys by which elements will be accessed in it. Taking inspiration from checkers' stored games use of prefixes and their use in the GetAllStoredGame function, prepare prefix keys for the values in a new types/key\_candidate.go file:



Copy

const (

CandidateKeyPrefix = "Candidate/value/"

)

// CandidateKey returns the store key to retrieve a Candidate from the index field

// It is not used but is here to remind where values are stored

func CandidateKey(address []byte) []byte {

var key []byte

prefixBytes := []byte(CandidateKeyPrefix)

key = append(key, prefixBytes...)

key = append(key, address...)

return key

}

x /

leaderboard /

types /

key\_candidate.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-handling/x/leaderboard/types/key_candidate.go" \l "L9-L23" \t "_blank)

[#Copy link](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#use-the-candidate-store) Use the candidate store

Now you can add the functions that will use the transient store at each update and on EndBlock. Add a new keeper/candidate.go file with:



Copy

func (k Keeper) SetCandidate(ctx sdk.Context, candidate types.Candidate) {

candidateStore := prefix.NewStore(ctx.TransientStore(k.tKey), []byte(types.CandidateKeyPrefix))

candidateBytes := k.cdc.MustMarshal(&candidate)

candidateStore.Set(candidate.Address, candidateBytes)

}

x /

leaderboard /

keeper /

candidate.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-handling/x/leaderboard/keeper/candidate.go" \l "L9-L13" \t "_blank)

This function saves the candidate at its address. Already having []byte Address in the Candidate object proves useful. This also means that if there are two updates in one block for a single player only the second update is recorded. In the case of a game that has only increasing scores, this is okay.

Next, taking inspiration from StoredGame again, add a function to get all candidates with an iterator:



Copy

func (k Keeper) GetAllCandidates(ctx sdk.Context) (candidates []types.Candidate) {

candidateStore := prefix.NewStore(ctx.TransientStore(k.tKey), []byte(types.CandidateKeyPrefix))

iterator := sdk.KVStorePrefixIterator(candidateStore, []byte{})

defer iterator.Close()

for ; iterator.Valid(); iterator.Next() {

var candidate types.Candidate

k.cdc.MustUnmarshal(iterator.Value(), &candidate)

candidates = append(candidates, candidate)

}

return

}

x /

leaderboard /

keeper /

candidate.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-handling/x/leaderboard/keeper/candidate.go" \l "L15-L28" \t "_blank)

This gets all candidates. There may be many, but not so many that it grinds the application. After all, it only gets all that was put during the block itself.

[#Copy link](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#leaderboard-handling) Leaderboard handling

You have created the leaderboard helper functions and the function to get all candidates. You can now update the leaderboard. This takes place in EndBlock.

First, in a separate file, add one function to the keeper:



Copy

func (k Keeper) CollectSortAndClipLeaderboard(ctx sdk.Context) {

leaderboard := k.GetLeaderboard(ctx)

updated := types.AddCandidatesAtNow(leaderboard.Winners, ctx.BlockTime(), k.GetAllCandidates(ctx))

params := k.GetParams(ctx)

if params.Length < uint64(len(updated)) {

updated = updated[:params.Length]

}

leaderboard.Winners = updated

k.SetLeaderboard(ctx, leaderboard)

}

x /

leaderboard /

keeper /

end\_block\_leaderboard\_handler.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-handling/x/leaderboard/keeper/end_block_leaderboard_handler.go" \l "L10-L17" \t "_blank)

This function gets the candidates from the transient store and the leaderboard from the regular store, adds the candidates, clips the array to the maximum length found in Params, and saves the updated leaderboard back in storage.

This means that the leaderboard will be unmarshalled and marshalled only once per block.

Next, make sure it is called from EndBlock. In module.go:



Copy

- func (am AppModule) EndBlock(\_ sdk.Context, \_ abci.RequestEndBlock) []abci.ValidatorUpdate {

+ func (am AppModule) EndBlock(ctx sdk.Context, \_ abci.RequestEndBlock) []abci.ValidatorUpdate {

+ am.keeper.CollectSortAndClipLeaderboard(ctx)

return []abci.ValidatorUpdate{}

}

x /

leaderboard /

module.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-handling/x/leaderboard/module.go" \l "L173-L174" \t "_blank)

If Ignite did it right, app.go has [already set up (opens new window)↗](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-handling/app/app.go#L523) the leaderboard module to be called on EndBlock.

Your leaderboard will now be updated and saved on an on-going basis as part of your v2 blockchain. However, so far, you have put **nothing** in the transient store.

[#Copy link](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#hook-infrastructure-for-candidates) Hook infrastructure for candidates

To populate candidate winners in your transient store, you are going to *listen* to PlayerInfo updates emitted from the checkers module:

* This will avoid tight coupling between the modules.
* The checkers module will not care whether there is a listener or not.
* It will be the duty of app.go to hook the leaderboard's listener to the checkers emitter.
* To reduce the dependency of the leaderboard module on elements of the checkers module, you are going to restrict to a single file.

With the Cosmos SDK, hooks are a design pattern so you have to code them.

[#Copy link](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#on-the-checkers-module) On the checkers module

Add the hooks interface to the checkers module. First as an expected interface:



Copy

type CheckersHooks interface {

AfterPlayerInfoChanged(ctx sdk.Context, playerInfo PlayerInfo)

}

x /

checkers /

types /

expected\_keepers.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-handling/x/checkers/types/expected_keepers.go" \l "L25-L27" \t "_blank)

Here you can imagine you could add functions for all sorts of updates coming from checkers. But for the sake of the exercise keep it simple.

Then, taking inspiration from the [governance module's hooks (opens new window)↗](https://github.com/cosmos/cosmos-sdk/blob/v0.45.4/x/gov/types/hooks.go#L7-L20), define a convenience multi hook that can accommodate multiple listeners:



Copy

var \_ CheckersHooks = MultiCheckersHooks{}

type MultiCheckersHooks []CheckersHooks

func NewMultiCheckersHooks(hooks ...CheckersHooks) MultiCheckersHooks {

return hooks

}

func (h MultiCheckersHooks) AfterPlayerInfoChanged(ctx sdk.Context, playerInfo PlayerInfo) {

for i := range h {

h[i].AfterPlayerInfoChanged(ctx, playerInfo)

}

}

x /

checkers /

types /

hooks.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-handling/x/checkers/types/hooks.go" \l "L7-L19" \t "_blank)

Expose this hooks interface via the checkers keeper:



Copy

type (

Keeper struct {

bank types.BankEscrowKeeper

+ hooks types.CheckersHooks

cdc codec.BinaryCodec

...

}

)

x /

checkers /

keeper /

keeper.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-handling/x/checkers/keeper/keeper.go" \l "L17" \t "_blank)

And a function to set it:



Copy

func (keeper \*Keeper) SetHooks(hooks types.CheckersHooks) \*Keeper {

if keeper.hooks != nil {

panic("cannot set checkers hooks twice")

}

keeper.hooks = hooks

return keeper

}

x /

checkers /

keeper /

keeper.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-handling/x/checkers/keeper/keeper.go" \l "L47-L55" \t "_blank)

Having a function to set the hooks is advised, as that allows you to collect the listeners you need without worrying about the order of creation of other keepers.

With the hooks structure in place, you must have your checkers code call it. The best place for that is precisely where it is updated and saved:



Copy

func mustAddDeltaGameResultToPlayer(

...

) (playerInfo types.PlayerInfo) {

...

k.SetPlayerInfo(ctx, playerInfo)

+ if k.hooks != nil {

+ k.hooks.AfterPlayerInfoChanged(ctx, playerInfo)

+ }

return playerInfo

}

x /

checkers /

keeper /

player\_info\_handler.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-handling/x/checkers/keeper/player_info_handler.go" \l "L32-L34" \t "_blank)



Remember that the hook is named AfterPlayerInfoChanged, not *for use by the leaderboard*. Therefore you should also emit when there is a change that you know is going to be discarded by the leaderboard.   
  
It verifies != nil to make sure it does not panic if there are no listeners, which is a legitimate situation.

The checkers module is now ready with regards to the hooks.

[#Copy link](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#on-the-leaderboard-module) On the leaderboard module

In your keeper, define a generic checkers hook listener. In a new keeper/hooks.go file, put a simple:



Copy

type Hooks struct {

k Keeper

}

func (k Keeper) Hooks() Hooks { return Hooks{k} }

x /

leaderboard /

keeper /

hooks.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-handling/x/leaderboard/keeper/hooks.go" \l "L3-L7" \t "_blank)

Then, so as to keep the dependency on checkers' types in as few files as possible, encapsulate the conversion knowledge in a new types/leaderboard\_checkers.go:



Copy

func MakeCandidateFromPlayerInfo(playerInfo checkerstypes.PlayerInfo) (candidate Candidate, err error) {

address, err := sdk.AccAddressFromBech32(playerInfo.Index)

if err != nil {

return candidate, sdkerrors.Wrapf(err, "Could not parse address from playerInfo %s", playerInfo.Index)

}

return Candidate{

Address: address,

WonCount: playerInfo.WonCount,

}, nil

}

x /

leaderboard /

types /

leaderboard\_checkers.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-handling/x/leaderboard/types/leaderboard_checkers.go" \l "L9-L18" \t "_blank)

Now encapsulate the handling in a new keeper/hooks\_checkers.go file:



Copy

var \_ checkerstypes.CheckersHooks = Hooks{}

func (h Hooks) AfterPlayerInfoChanged(ctx sdk.Context, playerInfo checkerstypes.PlayerInfo) {

candidate, err := types.MakeCandidateFromPlayerInfo(playerInfo)

if err != nil {

panic(fmt.Sprintf("%v", err))

}

if candidate.WonCount < 1 {

return

}

h.k.SetCandidate(ctx, candidate)

}

x /

leaderboard /

keeper /

hooks\_checkers.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-handling/x/leaderboard/keeper/hooks_checkers.go" \l "L11-L22" \t "_blank)

As you can see, this takes the new information and puts it into the transient store *only if it is worth doing so*.



If your leaderboard hooks listener was set to listen from more than one module, you would add a new hooks\_othermodule.go file that only concerns itself with that other module.

The leaderboard handling is now complete.

[#Copy link](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#on-app-go) On app.go

All app.go has to do is call checkers' SetHooks with the leaderboard's listener after all keepers have been created:



Copy

leaderboardModule := leaderboardmodule.NewAppModule(appCodec, app.LeaderboardKeeper, app.AccountKeeper, app.BankKeeper)

+ app.CheckersKeeper = \*app.CheckersKeeper.SetHooks(

+ checkersmoduletypes.NewMultiCheckersHooks(

+ app.LeaderboardKeeper.Hooks(),

+ ),

+ )

// this line is used by starport scaffolding # stargate/app/keeperDefinition

app /

app.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-handling/app/app.go" \l "L427-L431" \t "_blank)

Note how app.CheckersKeeper is replaced. This means that you need to move the checkers module line below:



Copy

- checkersModule := checkersmodule.NewAppModule(appCodec, app.CheckersKeeper, app.AccountKeeper, app.BankKeeper)

app.LeaderboardKeeper = \*leaderboardmodulekeeper.NewKeeper

...

app.CheckersKeeper = \*app.CheckersKeeper.SetHooks(

...

)

+ checkersModule := checkersmodule.NewAppModule(appCodec, app.CheckersKeeper, app.AccountKeeper, app.BankKeeper)

// this line is used by starport scaffolding # stargate/app/keeperDefinition

app /

app.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-handling/app/app.go" \l "L433" \t "_blank)

If you forgot to do so, the module would be created with the *hook-less* keeper.

[#Copy link](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#unit-tests) Unit tests

After all these changes, it is worthwhile adding tests.

Just like you did for the checkers module, you can add valid addresses to be reused elsewhere in a new file:



Copy

const (

Alice = "cosmos1jmjfq0tplp9tmx4v9uemw72y4d2wa5nr3xn9d3"

Bob = "cosmos1xyxs3skf3f4jfqeuv89yyaqvjc6lffavxqhc8g"

Carol = "cosmos1e0w5t53nrq7p66fye6c8p0ynyhf6y24l4yuxd7"

)

x /

leaderboard /

testutil /

constants.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-objects/x/leaderboard/testutil/constants.go" \l "L3-L7" \t "_blank)

[#Copy link](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#candidate-unit-tests) Candidate unit tests

You added a new Candidate type and helper functions on it. You can test that they work as expected. Add a new leaderboard\_test.go file. No need to overdo it:



Copy

const (

alice = testutil.Alice

)

func TestCandidateGetWinnerAtTime(t \*testing.T) {

now := time.Now()

timestamp := now.Unix()

aliceAddress, err := sdk.AccAddressFromBech32(alice)

require.Nil(t, err)

candidate := types.Candidate{

Address: aliceAddress,

WonCount: 23,

}

winner := candidate.GetWinnerAtTime(now)

require.EqualValues(t, types.Winner{

Address: alice,

WonCount: 23,

AddedAt: uint64(timestamp),

}, winner)

}

x /

leaderboard /

types /

leaderboard\_test.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-objects/x/leaderboard/types/leaderboard_test.go" \l "L13-L32" \t "_blank)

[#Copy link](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#leaderboard-helper-unit-tests) Leaderboard helper unit tests

Start by adding tests that confirm that the sorting of the leaderboard's winners works as expected. Here an array of test cases is a good choice:



Copy

func TestSortWinners(t \*testing.T) {

tests := []struct {

name string

unsorted []types.Winner

sorted []types.Winner

}{

{

name: "sort empty",

unsorted: []types.Winner{},

sorted: []types.Winner{},

},

{

name: "sort unique",

unsorted: []types.Winner{

{

Address: alice,

WonCount: 2,

AddedAt: 1000,

},

},

sorted: []types.Winner{

{

Address: alice,

WonCount: 2,

AddedAt: 1000,

},

},

},

... // More test cases

}

for \_, tt := range tests {

t.Run(tt.name, func(t \*testing.T) {

leaderboard := types.Leaderboard{

Winners: tt.unsorted,

}

leaderboard.SortWinners()

sorted := leaderboard.Winners

require.Equal(t, len(tt.sorted), len(sorted))

require.EqualValues(t, tt.sorted, sorted)

})

}

}

x /

leaderboard /

types /

leaderboard\_test.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-handling/x/leaderboard/types/leaderboard_test.go" \l "L35-L156" \t "_blank)

With that done, you can confirm that the updating or addition of new player info to the leaderboard works as expected, again with an array of test cases:



Copy

func TestAddCandidatesAtNow(t \*testing.T) {

aliceAdd, err := sdk.AccAddressFromBech32(alice)

require.Nil(t, err)

bobAdd, err := sdk.AccAddressFromBech32(bob)

require.Nil(t, err)

tests := []struct {

name string

sorted []types.Winner

candidates []types.Candidate

now int64

expected []types.Winner

}{

{

name: "add to empty",

sorted: []types.Winner{},

candidates: []types.Candidate{{

Address: aliceAdd,

WonCount: 2,

}},

now: 1000,

expected: []types.Winner{

{

Address: alice,

WonCount: 2,

AddedAt: 1000,

},

},

},

... // More test cases

}

for \_, tt := range tests {

t.Run(tt.name, func(t \*testing.T) {

actual := types.AddCandidatesAtNow(tt.sorted, time.Unix(tt.now, 0), tt.candidates)

require.Equal(t, len(tt.expected), len(actual))

require.EqualValues(t, tt.expected, actual)

require.NoError(t, types.Leaderboard{Winners: actual}.Validate())

})

}

}

x /

leaderboard /

types /

leaderboard\_test.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-handling/x/leaderboard/types/leaderboard_test.go" \l "L158-L433" \t "_blank)

[#Copy link](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#candidate-lifecycle-unit-tests) Candidate lifecycle unit tests

You added functions to set and get candidates from the transient store. You ought to add unit tests to confirm this works as expected.

First, you need to make sure that your test keeper has a valid transient store:



Copy

func LeaderboardKeeper(t testing.TB) (\*keeper.Keeper, sdk.Context) {

storeKey := sdk.NewKVStoreKey(types.StoreKey)

memStoreKey := storetypes.NewMemoryStoreKey(types.MemStoreKey)

+ tStoreKey := storetypes.NewTransientStoreKey(types.TStoreKey)

...

stateStore.MountStoreWithDB(memStoreKey, sdk.StoreTypeMemory, nil)

+ stateStore.MountStoreWithDB(tStoreKey, sdk.StoreTypeTransient, nil)

require.NoError(t, stateStore.LoadLatestVersion())

...

k := keeper.NewKeeper(

cdc,

storeKey,

memStoreKey,

+ tStoreKey,

paramsSubspace,

)

...

}

testutil /

keeper /

leaderboard.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-handling/testutil/keeper/leaderboard.go" \l "L23-L45" \t "_blank)

With this preparation, you can add simple tests. For example, that you get back [one candidate (opens new window)↗](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-handling/x/leaderboard/keeper/candidate_test.go#L20-L39) when there is one, or three when there are three:



Copy

func TestSetAndGetThreeCandidates(t \*testing.T) {

keeper, ctx := keepertest.LeaderboardKeeper(t)

aliceAddress, err := sdk.AccAddressFromBech32(alice)

require.Nil(t, err)

bobAddress, err := sdk.AccAddressFromBech32(bob)

require.Nil(t, err)

carolAddress, err := sdk.AccAddressFromBech32(carol)

require.Nil(t, err)

keeper.SetCandidate(ctx, types.Candidate{

Address: aliceAddress,

WonCount: 12,

})

keeper.SetCandidate(ctx, types.Candidate{

Address: bobAddress,

WonCount: 34,

})

keeper.SetCandidate(ctx, types.Candidate{

Address: carolAddress,

WonCount: 56,

})

candidates := keeper.GetAllCandidates(ctx)

require.Len(t, candidates, 3)

sort.SliceStable(candidates[:], func(i, j int) bool {

return candidates[i].WonCount < candidates[j].WonCount

})

require.Equal(t,

[]types.Candidate{

{Address: aliceAddress, WonCount: 12},

{Address: bobAddress, WonCount: 34},

{Address: carolAddress, WonCount: 56},

},

candidates,

)

}

x /

leaderboard /

keeper /

candidate\_test.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-handling/x/leaderboard/keeper/candidate_test.go" \l "L41-L76" \t "_blank)



Note the small hack where the received candidates are sorted by WonCount. The GetAllCandidates function does not ensure an order, so to be able to easily use require.Equal an ordering was used.

[#Copy link](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#leaderboard-handling-unit-tests) Leaderboard handling unit tests

You can verify that the leaderboard is updated when the keeper.CollectSortAndClipLeaderboard function is called.

To change the context time, you can use the SDK context's WithBlockTime function. For instance, test when a single candidate is [added between two (opens new window)↗](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-handling/x/leaderboard/keeper/end_block_leaderboard_handler_test.go#L13-L41) existing winners. Or when one candidate replaces its lower score and another enters the leaderboard for the first time:



Copy

func TestOnePlayerAddedAndOneUpdatedToLeaderboard(t \*testing.T) {

keeper, ctx := keepertest.LeaderboardKeeper(t)

keeper.SetLeaderboard(ctx, types.Leaderboard{

Winners: []types.Winner{

{Address: alice, WonCount: 12, AddedAt: 999},

{Address: bob, WonCount: 10, AddedAt: 999},

},

})

bobAddress, err := sdk.AccAddressFromBech32(bob)

require.Nil(t, err)

carolAddress, err := sdk.AccAddressFromBech32(carol)

require.Nil(t, err)

keeper.SetCandidate(ctx, types.Candidate{

Address: bobAddress,

WonCount: 13,

})

keeper.SetCandidate(ctx, types.Candidate{

Address: carolAddress,

WonCount: 12,

})

bobTime := time.Unix(1000, 0)

keeper.CollectSortAndClipLeaderboard(ctx.WithBlockTime(bobTime))

leaderboard := keeper.GetLeaderboard(ctx)

require.Len(t, leaderboard.Winners, 3)

require.Equal(t,

[]types.Winner{

{Address: bob, WonCount: 13, AddedAt: 1000},

{Address: carol, WonCount: 12, AddedAt: 1000},

{Address: alice, WonCount: 12, AddedAt: 999},

},

leaderboard.Winners,

)

}

x /

leaderboard /

keeper /

end\_block\_leaderboard\_handler\_test.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-handling/x/leaderboard/keeper/end_block_leaderboard_handler_test.go" \l "L43-L76" \t "_blank)

Where:

* You put a leaderboard in storage.
* Put candidates in the transient storage.
* Call the collection of candidates.
* Confirm the new leaderboard order and values.

You can also add a test that confirms the leaderboard is clipped at the maximum length:



Copy

func TestOnePlayerKicksPlayerOutOfLeaderboard(t \*testing.T) {

keeper, ctx := keepertest.LeaderboardKeeper(t)

keeper.SetLeaderboard(ctx, types.Leaderboard{

Winners: []types.Winner{

{Address: alice, WonCount: 12, AddedAt: 999},

{Address: bob, WonCount: 10, AddedAt: 999},

},

})

params := keeper.GetParams(ctx)

params.Length = 2

keeper.SetParams(ctx, params)

carolAddress, err := sdk.AccAddressFromBech32(carol)

require.Nil(t, err)

keeper.SetCandidate(ctx, types.Candidate{

Address: carolAddress,

WonCount: 11,

})

carolTime := time.Unix(1000, 0)

keeper.CollectSortAndClipLeaderboard(ctx.WithBlockTime(carolTime))

leaderboard := keeper.GetLeaderboard(ctx)

require.Len(t, leaderboard.Winners, 2)

require.Equal(t,

[]types.Winner{

{Address: alice, WonCount: 12, AddedAt: 999},

{Address: carol, WonCount: 11, AddedAt: 1000},

},

leaderboard.Winners,

)

}

x /

leaderboard /

keeper /

end\_block\_leaderboard\_handler\_test.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-handling/x/leaderboard/keeper/end_block_leaderboard_handler_test.go" \l "L78-L108" \t "_blank)

Where carol kicked bob out of the leaderboard since its length was enforced at 2:

[#Copy link](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#hook-unit-tests-on-leaderboard) Hook unit tests on leaderboard

Moving to the hooks *on the leaderboard module's side*, you want to confirm that candidates are added to the transient store when the keeper receives a new update:



Copy

func TestOneCandidateAdded(t \*testing.T) {

keeper, ctx := keepertest.LeaderboardKeeper(t)

keeper.Hooks().AfterPlayerInfoChanged(ctx, checkerstypes.PlayerInfo{

Index: alice,

WonCount: 12,

LostCount: 13,

ForfeitedCount: 14,

})

aliceAddress, err := sdk.AccAddressFromBech32(alice)

require.Nil(t, err)

candidates := keeper.GetAllCandidates(ctx)

require.Len(t, candidates, 1)

require.Equal(t,

types.Candidate{Address: aliceAddress, WonCount: 12},

candidates[0],

)

}

x /

leaderboard /

keeper /

hooks\_checkers\_test.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-handling/x/leaderboard/keeper/hooks_checkers_test.go" \l "L14-L31" \t "_blank)

Confirm also that it overwrites when it receives [an update for the same address (opens new window)↗](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-handling/x/leaderboard/keeper/hooks_checkers_test.go#L33-L56), or adds a second candidate [alongside an existing one (opens new window)↗](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-handling/x/leaderboard/keeper/hooks_checkers_test.go#L58-L84).

[#Copy link](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#hook-unit-tests-on-checkers) Hook unit tests on checkers

You introduced a new type, the MultiHook. You should test that it indeed distributes calls to the elements of the list. This calls for a mock of the CheckersHooks expected interface.

Run again your [existing script](https://ida.interchain.io/hands-on-exercise/2-ignite-cli-adv/6-payment-winning.html) that rebuilds all the mocks.

**Local**

**Docker**



Copy

$ make mock-expected-keepers

Copy

$ docker run --rm -it \

-v $(pwd):/checkers \

-w /checkers \

checkers\_i \

make mock-expected-keepers

With that, you can add a test that confirms a multihook with two hooks calls both in order:



Copy

func TestMultiHookCallsThem(t \*testing.T) {

ctrl := gomock.NewController(t)

defer ctrl.Finish()

hook1 := testutil.NewMockCheckersHooks(ctrl)

hook2 := testutil.NewMockCheckersHooks(ctrl)

call1 := hook1.EXPECT().AfterPlayerInfoChanged(gomock.Any(), types.PlayerInfo{

Index: "alice",

WonCount: 1,

LostCount: 2,

ForfeitedCount: 3,

}).Times(1)

hook2.EXPECT().AfterPlayerInfoChanged(gomock.Any(), types.PlayerInfo{

Index: "alice",

WonCount: 1,

LostCount: 2,

ForfeitedCount: 3,

}).Times(1).After(call1)

multi := types.NewMultiCheckersHooks(hook1, hook2)

multi.AfterPlayerInfoChanged(sdk.NewContext(nil, tmproto.Header{}, false, nil), types.PlayerInfo{

Index: "alice",

WonCount: 1,

LostCount: 2,

ForfeitedCount: 3,

})

}

x /

checkers /

types /

hooks\_test.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-handling/x/checkers/types/hooks_test.go" \l "L13-L38" \t "_blank)

Your existing checkers keeper tests should still be passing.

However, there is a small difficulty that would not surface immediately: when you set the hooks after the msgServer has been created, because it takes a keeper instance and not a pointer, the msgServer is created with the *old* keeper (the one before the hooks were set).

Therefore, add a setup function that encapsulates the knowledge to circumvent this difficulty:



Copy

func setupMsgServerWithOneGameForPlayMoveAndHooks(t testing.TB) (types.MsgServer, keeper.Keeper, context.Context,

\*gomock.Controller, \*testutil.MockBankEscrowKeeper, \*testutil.MockCheckersHooks) {

msgServer, k, context, ctrl, escrow := setupMsgServerWithOneGameForPlayMove(t)

hookMock := testutil.NewMockCheckersHooks(ctrl)

k.SetHooks(hookMock)

msgServer = keeper.NewMsgServerImpl(k)

return msgServer, k, context, ctrl, escrow, hookMock

}

x /

checkers /

keeper /

msg\_server\_play\_move\_test.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-handling/x/checkers/keeper/msg_server_play_move_test.go" \l "L35-L42" \t "_blank)

You can now add a test that confirms that a game just played does not trigger a call to the hooks:



Copy

func TestPlayerInfoNoHookOnNoWinner(t \*testing.T) {

msgServer, keeper, context, ctrl, escrow, \_ := setupMsgServerWithOneGameForPlayMoveAndHooks(t)

ctx := sdk.UnwrapSDKContext(context)

defer ctrl.Finish()

escrow.ExpectAny(context)

keeper.SetPlayerInfo(ctx, types.PlayerInfo{

Index: bob,

})

keeper.SetPlayerInfo(ctx, types.PlayerInfo{

Index: carol,

})

msgServer.PlayMove(context, &types.MsgPlayMove{

Creator: bob,

GameIndex: "1",

FromX: 1,

FromY: 2,

ToX: 2,

ToY: 3,

})

}

x /

checkers /

keeper /

msg\_server\_play\_move\_test.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-handling/x/checkers/keeper/msg_server_play_move_test.go" \l "L607-L626" \t "_blank)

A more interesting addition is the confirmation that a listener is being called when a game is [forfeited (opens new window)↗](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-handling/x/checkers/keeper/end_block_server_game_test.go#L684-L737) or won:



Copy

func TestCompleteGameCallsHook(t \*testing.T) {

msgServer, keeper, context, ctrl, escrow, hookMock := setupMsgServerWithOneGameForPlayMoveAndHooks(t)

ctx := sdk.UnwrapSDKContext(context)

defer ctrl.Finish()

escrow.ExpectAny(context)

bobCall := hookMock.EXPECT().AfterPlayerInfoChanged(ctx, types.PlayerInfo{

Index: bob,

WonCount: 2,

LostCount: 2,

ForfeitedCount: 3,

}).Times(1)

hookMock.EXPECT().AfterPlayerInfoChanged(ctx, types.PlayerInfo{

Index: carol,

WonCount: 4,

LostCount: 6,

ForfeitedCount: 6,

}).Times(1).After(bobCall)

keeper.SetPlayerInfo(ctx, types.PlayerInfo{

Index: bob,

WonCount: 1,

LostCount: 2,

ForfeitedCount: 3,

})

keeper.SetPlayerInfo(ctx, types.PlayerInfo{

Index: carol,

WonCount: 4,

LostCount: 5,

ForfeitedCount: 6,

})

testutil.PlayAllMoves(t, msgServer, context, "1", bob, carol, testutil.Game1Moves)

}

x /

checkers /

keeper /

msg\_server\_play\_move\_winner\_test.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-handling/x/checkers/keeper/msg_server_play_move_winner_test.go" \l "L133-L165" \t "_blank)

[#Copy link](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#integration-tests) Integration tests

To further confirm that your code is working correctly you can add integration tests. Since it starts an app, the hooks are already set up.

You could decide to piggy-back on the existing "checkers" integration tests. However, for the sake of clarity, create a separate folder:



Copy

$ mkdir -p tests/integration/leaderboard/keeper

Copy the integration test suite from the checkers integration tests, with adjusted imports and others, minus all the balances and denoms. Keep the msgServer, as that is the one that receives messages:



Copy

const (

alice = testutil.Alice

bob = testutil.Bob

carol = testutil.Carol

)

type IntegrationTestSuite struct {

suite.Suite

app \*checkersapp.App

msgServer checkerstypes.MsgServer

ctx sdk.Context

queryClient types.QueryClient

}

func TestLeaderboardKeeperTestSuite(t \*testing.T) {

suite.Run(t, new(IntegrationTestSuite))

}

func (suite \*IntegrationTestSuite) SetupTest() {

app := checkersapp.Setup(false)

ctx := app.BaseApp.NewContext(false, tmproto.Header{Time: time.Now()})

app.AccountKeeper.SetParams(ctx, authtypes.DefaultParams())

app.BankKeeper.SetParams(ctx, banktypes.DefaultParams())

queryHelper := baseapp.NewQueryServerTestHelper(ctx, app.InterfaceRegistry())

types.RegisterQueryServer(queryHelper, app.LeaderboardKeeper)

queryClient := types.NewQueryClient(queryHelper)

suite.app = app

suite.msgServer = checkerskeeper.NewMsgServerImpl(app.CheckersKeeper)

suite.ctx = ctx

suite.queryClient = queryClient

}

You can confirm the leaderboard is called when a game is won:



Copy

func (suite \*IntegrationTestSuite) setupSuiteWithOneGameForPlayMove() {

goCtx := sdk.WrapSDKContext(suite.ctx)

suite.msgServer.CreateGame(goCtx, &checkerstypes.MsgCreateGame{

Creator: alice,

Black: bob,

Red: carol,

Wager: 0,

Denom: "stake",

})

}

func (suite \*IntegrationTestSuite) TestPlayMoveToWinnerAddedToLeaderboard() {

suite.setupSuiteWithOneGameForPlayMove()

suite.app.CheckersKeeper.SetPlayerInfo(suite.ctx, checkerstypes.PlayerInfo{

Index: alice, WonCount: 10,

})

suite.app.CheckersKeeper.SetPlayerInfo(suite.ctx, checkerstypes.PlayerInfo{

Index: bob, WonCount: 10,

})

suite.app.LeaderboardKeeper.SetLeaderboard(suite.ctx, leaderboardtypes.Leaderboard{

Winners: []leaderboardtypes.Winner{

{Address: alice, WonCount: 10, AddedAt: 1000},

{Address: bob, WonCount: 10, AddedAt: 999},

},

})

testutil.PlayAllMoves(suite.T(), suite.msgServer, sdk.WrapSDKContext(suite.ctx), "1", bob, carol, testutil.Game1Moves)

suite.app.LeaderboardKeeper.CollectSortAndClipLeaderboard(suite.ctx)

leaderboard := suite.app.LeaderboardKeeper.GetLeaderboard(suite.ctx)

suite.Require().EqualValues(

[]leaderboardtypes.Winner{

{Address: bob, WonCount: 11, AddedAt: uint64(suite.ctx.BlockTime().Unix())},

{Address: alice, WonCount: 10, AddedAt: 1000},

},

leaderboard.Winners)

}

tests /

integration /

... /

keeper /

msg\_server\_play\_move\_test.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-handling/tests/integration/leaderboard/keeper/msg_server_play_move_test.go" \t "_blank)

Or when a game is expired:



Copy

func (suite \*IntegrationTestSuite) TestForfeitPlayedTwiceCalledHooks() {

suite.setupSuiteWithOneGameForPlayMove()

goCtx := sdk.WrapSDKContext(suite.ctx)

suite.msgServer.PlayMove(goCtx, &types.MsgPlayMove{

Creator: bob,

GameIndex: "1",

FromX: 1,

FromY: 2,

ToX: 2,

ToY: 3,

})

suite.msgServer.PlayMove(goCtx, &types.MsgPlayMove{

Creator: carol,

GameIndex: "1",

FromX: 0,

FromY: 5,

ToX: 1,

ToY: 4,

})

keeper := suite.app.CheckersKeeper

keeper.SetPlayerInfo(suite.ctx, types.PlayerInfo{

Index: bob, WonCount: 10,

})

keeper.SetPlayerInfo(suite.ctx, types.PlayerInfo{

Index: carol, WonCount: 10,

})

suite.app.LeaderboardKeeper.SetLeaderboard(suite.ctx, leaderboardtypes.Leaderboard{

Winners: []leaderboardtypes.Winner{

{Address: bob, WonCount: 10, AddedAt: 1000},

{Address: carol, WonCount: 10, AddedAt: 999},

},

})

game1, found := keeper.GetStoredGame(suite.ctx, "1")

suite.Require().True(found)

oldDeadline := types.FormatDeadline(suite.ctx.BlockTime().Add(time.Duration(-1)))

game1.Deadline = oldDeadline

keeper.SetStoredGame(suite.ctx, game1)

keeper.ForfeitExpiredGames(goCtx)

suite.app.LeaderboardKeeper.CollectSortAndClipLeaderboard(suite.ctx)

leaderboard := suite.app.LeaderboardKeeper.GetLeaderboard(suite.ctx)

suite.Require().EqualValues(

[]leaderboardtypes.Winner{

{Address: carol, WonCount: 11, AddedAt: uint64(suite.ctx.BlockTime().Unix())},

{Address: bob, WonCount: 10, AddedAt: 1000},

},

leaderboard.Winners)

}

Expand



tests /

integration /

... /

keeper /

end\_block\_server\_game\_test.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/leaderboard-handling/tests/integration/leaderboard/keeper/end_block_server_game_test.go" \t "_blank)

Note how you have to call both *end blockers* because there are actually no blocks being produced. This recalls what you did previously when integration-testing the game forfeit.

This completes your checkers v2 chain. If you were to start it anew as is, it would work. If you want to see how you would migrate your blockchain if it were running v1.1, jump straight to the [next section](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/4-migration-leaderboard.html).

[#Copy link](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#interact-via-the-cli) Interact via the CLI

Your v2 blockchain is fully functioning. It will work as long as you start it from scratch (i.e. you should not try to migrate).

You should already know your way around testing this way. The simplest way is to use Ignite:

**Local**

**Docker**



Copy

$ ignite chain serve --reset-once

Copy

$ docker network create checkers-net

$ docker run --rm -it \

-v $(pwd):/checkers -w /checkers \

-p 4500:4500 -p 26657:26657 \

--network checkers-net \

--name checkers \

checkers\_i \

ignite chain serve --reset-once

Use your CosmJS integration tests to run a full game:

**Local**

**Docker**



Copy

$ npm test --prefix client

Copy

$ docker run --rm -it \

-v $(pwd)/client:/client -w /client \

--network checkers-net \

--env RPC\_URL="http://checkers:26657" \

node:18.7-slim \

npm test

After that, you can query your leaderboard:

**Local**

**Docker**



Copy

$ checkersd query leaderboard show-leaderboard

Copy

$ docker exec -it \

checkers \

checkersd query leaderboard show-leaderboard

It should turn something like:



Copy

Leaderboard:

winners:

- addedAt: "1682373982"

address: cosmos1fx6qlxwteeqxgxwsw83wkf4s9fcnnwk8z86sql

wonCount: "1"

Congratulations, your leaderboard is functional!

If you used Docker, you can stop the container and remove the network:



Copy

$ docker network rm checkers-net

synopsis

To summarize, this section has explored:

* How to add a leaderboard as a module to an existing blockchain, and the characteristics that a good leaderboard should boast.
* How to keep modules loosely coupled, when possible, with the use of hooks.
* How to leverage the transient store to save data for use in EndBlock.
* How to reduce computations and overall blockchain burden by ordering the leaderboard only once per block, in EndBlock.
* Worthwhile unit tests, including recreating the mocks, and integration tests.

previous

[](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/2-migration-info.html)

**[Tally Player Info After Production](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/2-migration-info.html)**

up next

**[Migrate the Leaderboard Module After Production](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/4-migration-leaderboard.html)**

[[](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/4-migration-leaderboard.html)](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/4-migration-leaderboard.html)

Rate this Page

icon smile

icon meh

icon frown

Would you like to add a message?

Submit

Thank you for your Feedback!

[](https://ida.interchain.io/ida-course/discord-info.html)

On this page

[High level considerations](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#high-level-considerations)

[What you will do](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#what-you-will-do)

[New v2 module](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#new-v2-module)

[New v2 information](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#new-v2-information)

[Transient object](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#transient-object)

[Leaderboard helpers](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#leaderboard-helpers)

[Candidate Lifecycle](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#candidate-lifecycle)

[Prepare transient store](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#prepare-transient-store)

[Prepare candidate store keys](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#prepare-candidate-store-keys)

[Use the candidate store](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#use-the-candidate-store)

[Leaderboard handling](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#leaderboard-handling)

[Hook infrastructure for candidates](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#hook-infrastructure-for-candidates)

[On the checkers module](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#on-the-checkers-module)

[On the leaderboard module](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#on-the-leaderboard-module)

[On app.go](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#on-app-go)

[Unit tests](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#unit-tests)

[Candidate unit tests](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#candidate-unit-tests)

[Leaderboard helper unit tests](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#leaderboard-helper-unit-tests)

[Candidate lifecycle unit tests](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#candidate-lifecycle-unit-tests)

[Leaderboard handling unit tests](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#leaderboard-handling-unit-tests)

[Hook unit tests on leaderboard](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#hook-unit-tests-on-leaderboard)

[Hook unit tests on checkers](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#hook-unit-tests-on-checkers)

[Integration tests](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#integration-tests)

[Interact via the CLI](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html#interact-via-the-cli)

#### **Get Cosmos updates**

Unsubscribe at any time. [Privacy Policy↗](https://v1.cosmos.network/privacy)

     Next

Documentation

[Cosmos SDK](https://docs.cosmos.network/)[Cosmos Hub](https://hub.cosmos.network/)[CometBFT](https://docs.cometbft.com/)[IBC Protocol](https://ibc.cosmos.network/)

Community

[Interchain blog](https://blog.cosmos.network/)[Forum](https://forum.cosmos.network/)[Discord](https://discord.gg/cosmosnetwork)

Contributing

[Source code on GitHub](https://github.com/cosmos/sdk-tutorials)

[](https://ida.interchain.io/)

[Interchain Developer Academy](https://ida.interchain.io/)

**[](https://blog.cosmos.network/)[](https://twitter.com/cosmos)[](https://discord.gg/cosmosnetwork)[](https://www.linkedin.com/company/interchain-foundation/about/)[](https://reddit.com/r/cosmosnetwork)[](https://t.me/cosmosproject)[](https://www.youtube.com/c/CosmosProject)**



Dark mode

† This website is maintained by the Interchain Foundation (ICF). The contents and opinions of this website are those of the ICF. The ICF provides links to cryptocurrency exchanges as a service to the public. The ICF does not warrant that the information provided by these websites is correct, complete, and up-to-date. The ICF is not responsible for their content and expressly rejects any liability for damages of any kind resulting from the use, reference to, or reliance on any information contained within these websites.

Cosmos is a registered trademark of the [Interchain Foundation.](https://interchain.io/)[Privacy](https://v1.cosmos.network/privacy)