LAB

ONCHAIN PROGRAMMING

1. Install Solana CLI, Rust and Anchor

Solana CLI release

Rust 1.76

```
curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
```

Anchor 0.30

```
cargo install --git https://github.com/coral-xyz/anchor avm --locked --force
```

```
avm install latest
```

```
avm use latest
```

Yarn

```
npm i -g yarn
```

2. Make a new anchor project, and run the default tests

```
anchor init favorites

cd favorites

npm i

anchor test
```

3. Ensure `anchor test` runs without errors!

Error	Fix
package `solana-program v1.18.11` cannot be built... cargo build-sbf --version	cargo add solana-program@"=1.18.5"
error: no such command: `build-sbf`	`export PATH=~/.local/share/solana/install/active_release/bin:$PATH` Add this to your ~/.zshrc or ~/.bashrc file to make the change permanent.

Unable to get latest blockhash. Test validator does not look started.	Install Homebrew and use it to install GNU tar:
	`brew install gnu-tar`
	`export PATH=/opt/homebrew/opt/gnu-tar/libexec/gnubin:$PATH`
	Add this to your ~/.zshrc or ~/.bashrc file to make the change permanent.

No license field in package.json warning	Open package.json, add "license": "MIT" or "license": "UNLICENSED" depending on preferences.
Error: Unable to read keypair file	Add a keypair (just the array of numbers) to ~/.config/solana/id.json `solana-keygen new --no-bip39-passphrase`

4. Questions:

Does the anchor project compile without any warnings?

What is the name of the instruction handler?

The program has a ID, which is a pubkey.
Where does the secret key for this pubkey exist? Hint: it's in your project.

5. Import Anchor prelude, and tell Anchor what we will store in our PDA

programs/favorites/src/lib.rs

```rust
use anchor_lang::prelude::*;

// Our program's address! You don't need to change it.
// This matches the private key in the target/deploy directory
declare_id!("YOUR PROGRAM's PUBLIC KEY (KEEP THIS AS-IS)");

// Anchor accounts use 8 bytes to determine their type
pub const ANCHOR_DISCRIMINATOR_SIZE: usize = 8;
```

continued on next slide...

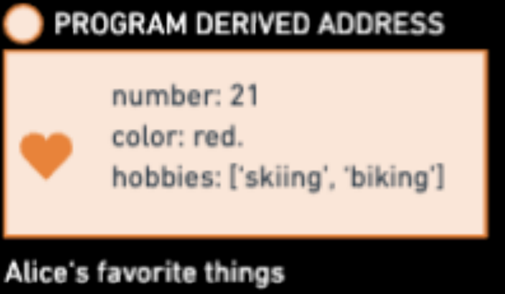
6. Make a Solana Program!

programs/favorites/src/lib.rs continued...

```rust
// A 'mod' is a Rust module. But the #[program] makes it into a Solana program!
#[program]
pub mod favorites {
    use super::*;

    // Our instruction handler! It sets the user's favorite number and color
    pub fn set_favorites(context: Context<SetFavorites>, number: u64, color: String, hobbies:
Vec<String>) -> Result<()> {
        let user_public_key = context.accounts.user.key();
        msg!("Greetings from {}", context.program_id);
        msg!(
            "User {user_public_key}'s favorite number is {number}, favorite color is: {color}",
        );

        msg!(
            "User's hobbies are: {:?}",
            hobbies
        );

        context.accounts.favorites.set_inner(Favorites {
            number,
            color,
            hobbies
        });
        Ok(())
    }

    // We can also add a get_favorites instruction handler to return the user's favorite number and color
}
```

continued on next slide...

7. Tell Anchor what will go inside our PDA



programs/favorites/src/lib.rs continued...

```rust
// What we will put inside the Favorites PDA
#[account]
#[derive(InitSpace)]
pub struct Favorites {
    pub number: u64,

    #[max_len(50)]
    pub color: String,

    #[max_len(5, 50)]
    pub hobbies: Vec<String>
}
```

continued on next slide...

8. Tell Anchor the accounts that must be provided

programs/favorites/src/lib.rs continued...

```
// When people call the set_favorites instruction, they will need to provide the
accounts that will be modifed. This keeps Solana fast!
#[derive(Accounts)]
pub struct SetFavorites<'info> {
    #[account(mut)]
    pub user: Signer<'info>,

    #[account(
        init,
        payer = user,
        space = ANCHOR_DISCRIMINATOR_SIZE + Favorites::INIT_SPACE,
        seeds=[b"favorites", user.key().as_ref()],
    bump)]
    pub favorites: Account<'info, Favorites>,

    pub system_program: Program<'info, System>,
}
```

Done! You have made a solana program

8. Build the project with

```
anchor build
```

Check your project compiles!

```
favorites % anchor build
    Finished release [optimized] target(s) in 0.07s
    Finished `test` profile [unoptimized + debuginfo] target(s) in 0.11s
     Running unittests src/lib.rs (/Users/mikemaccana/Code/solana/professional-
371dae)
🤠 /Users/mikemaccana/Code/solana/professional-education/labs/favorites
```

9. Make some tests part 1/4

Inside **tests/favorites.ts**, add:

```
import { assert } from "chai";
const web3 = anchor.web3;
```

Then replace the **'describe'** function:

```
describe("Favorites", () => {
  // Use the cluster and the keypair specified in Anchor.toml
  const provider = anchor.AnchorProvider.env();
  anchor.setProvider(provider);
  const user = (provider.wallet as anchor.Wallet).payer;
  const program = anchor.workspace.Favorites as Program<Favorites>;

  // You can skip this 'before' section if you're busy!
  // We don't need to airdrop if we're using the local cluster
  // because the local cluster gives us 85 billion dollars worth of SOL
  before(async () => {
    const balance = await provider.connection.getBalance(user.publicKey);
    const balanceInSOL = balance / web3.LAMPORTS_PER_SOL;
    const formattedBalance = new Intl.NumberFormat().format(balanceInSOL);
    console.log(`Balance: ${formattedBalance} SOL`);
  });
```

continued on next slide...

10. Make some tests part 2/4

continued from previous slide

```
it("Saves a user's favorites to the blockchain", async () => {
  // Here's what we want to write to the blockchain
  const favoriteNumber = new anchor.BN(23);
  const favoriteColor = "purple";
  const favoriteHobbies = ["skiing", "skydiving", "biking"];

  await program.methods
    .setFavorites(favoriteNumber, favoriteColor, favoriteHobbies)
    .signers([user])
    .rpc();

  // No check everything matches
  const favoritesPdaAndBump = web3.PublicKey.findProgramAddressSync(
    [Buffer.from("favorites"), user.publicKey.toBuffer()],
    program.programId
  );
  const favoritesPda = favoritesPdaAndBump[0];
  const dataFromPda = await program.account.favorites.fetch(favoritesPda);
  assert.equal(dataFromPda.color, favoriteColor);
  assert.equal(dataFromPda.number.toString(), favoriteNumber.toString());
  assert.deepEqual(dataFromPda.hobbies, favoriteHobbies);
});
```

continued on next slide...

11. Make some tests part 4/4

continued from previous slide

```
it("Doesn't let people write to favorites for other users", async () => {
  const someRandomGuy = anchor.web3.Keypair.generate();
  try {
    await program.methods
      .setFavorites(new anchor.BN(420), "red", ["being a dork"])
      .signers([someRandomGuy])
      .rpc();
  } catch (error) {
    const errorMessage = (error as Error).message;
    assert.isTrue(errorMessage.includes("unknown signer"));
  }
});
```

Close the 'describe' block:

```
});
```

You've finished making your anchor tests!

12. Run the tests with

```
anchor test
```

Ensure your tests pass!

```
Favorites
  ✔ Writes our favorites to the blockchain (201ms)


1 passing (203ms)
```

Lesson 6 feedback

Let us know what you think!