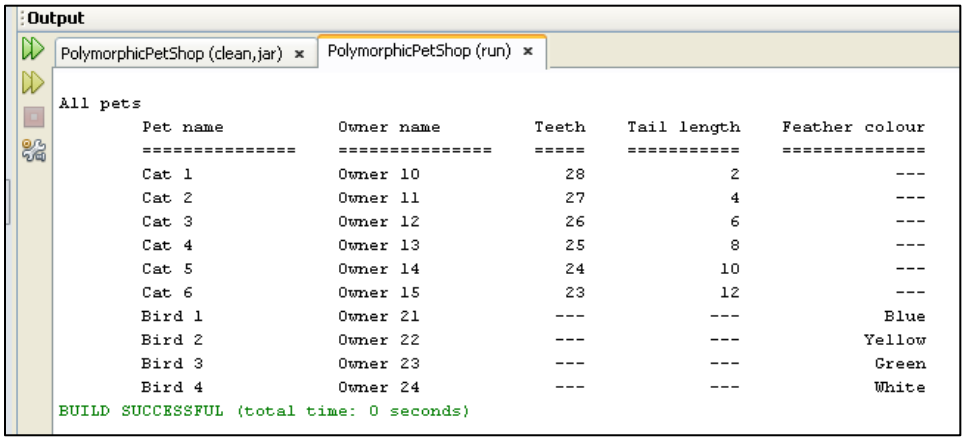
Practical session 2

This work should be completed before the next lecture.

Task 1: Pet shop with abstract `Pet`

Make a copy of your NetBeans project from Task 3 of Week 1.

Make the `Pet` class abstract. Correct any compilation errors that might be caused. The output should still look similar to this…



```
Output
PolymorphicPetShop (clean,jar)   PolymorphicPetShop (run)

All pets
        Pet name            Owner name          Teeth    Tail length    Feather colour
        ===============     ===============     =====    ===========    ==============
        Cat 1               Owner 10               28              2               ---
        Cat 2               Owner 11               27              4               ---
        Cat 3               Owner 12               26              6               ---
        Cat 4               Owner 13               25              8               ---
        Cat 5               Owner 14               24             10               ---
        Cat 6               Owner 15               23             12               ---
        Bird 1              Owner 21              ---            ---              Blue
        Bird 2              Owner 22              ---            ---            Yellow
        Bird 3              Owner 23              ---            ---             Green
        Bird 4              Owner 24              ---            ---             White
BUILD SUCCESSFUL (total time: 0 seconds)
```

Portfolio requirements:

- The NetBeans project for this completed task

Task 2: Pet shop with `equals()` and `hashCode()`

Make a copy of your NetBeans project from Task 1 and modify it as follows:

- In the `Pet` class, add two abstract methods to override the `equals()` and `hashCode()` methods.
- In the `Cat` class, override the `equals()` and `hashCode()` methods (you can use the "Insert code…" feature in NetBeans to generate them).
- In the `PetShopApplication` class, create another `Cat` object with the same data as one of the objects already created, and test to see if the two objects are equal. Output a message to confirm equality.
- In the `PetShopApplication` class, test two unequal `Cat` objects to see if they are equal. Output a message to confirm non-equality.

Portfolio requirements:

- The NetBeans project for this completed task

Task 3: Deck of Cards

In a new NetBeans project do the following:

- Create a `Card` class, that has two variables: Suit (Hearts, Clubs, Diamonds, Spades) and Rank (Ace, Two, Three, Four, … Jack, Queen, King). Use enums to represent Suit and Rank. Add suitable accessor (get) methods.
- Create a `Deck` class that has an array of 52 `Card` objects (one of each rank for each suit). Add an accessor method to return the array.
- Create a `CardApplication` class that creates a `Deck` object, gets, the array of `Card` objects, and outputs to the console window the entire deck of cards.

Portfolio requirements:

- The NetBeans project for this completed task

Task 4: Deck of Cards with local class

Make a copy of your NetBeans project from Task 3 and modify it as follows:

- In the `Deck` class, add a method that uses a local class to return an iterator for the array of `Card` objects (see lecture notes).
- Modify the `CardApplication` class so that it uses the iterator from the `Deck` object to output to the console window the entire deck of cards.

Portfolio requirements:

- The NetBeans project for this completed task