Noroff
School of technology
and digital media

Wednesday - Week 1

Classes

Classes

- At their simplest they are a “combination of variables, functions, and data structures.”
- Representations of things that exist.

Some Java Rules

- The file name must match the class name
- One class per file
- The package name (FQN) must match the path to the file

- This is HORRIBLE to work with in console/terminal
- SublimeText may fail us here!

Noroff
School of technology
and digital media

Wednesday - Week 1

Compiling with multiple source

Class Topics for Today

- Access modifiers
- Namespaces & Packages
- Constructors

Access Modifiers

- `public`
  - Available to anything that imports the class
- `protected`
  - Available to the current class and subclasses
- `private`
  - Available to the current class only

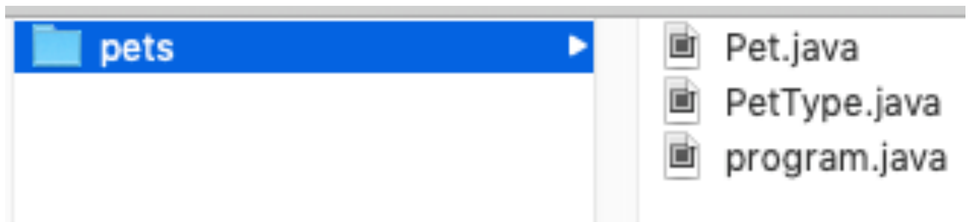
Namespaces & Packages

- Lets start with a simple example
- We want to build a program to ‘store’ pet info
- This will be in the “pets” package

Project structure

- I create three files (folders are important):

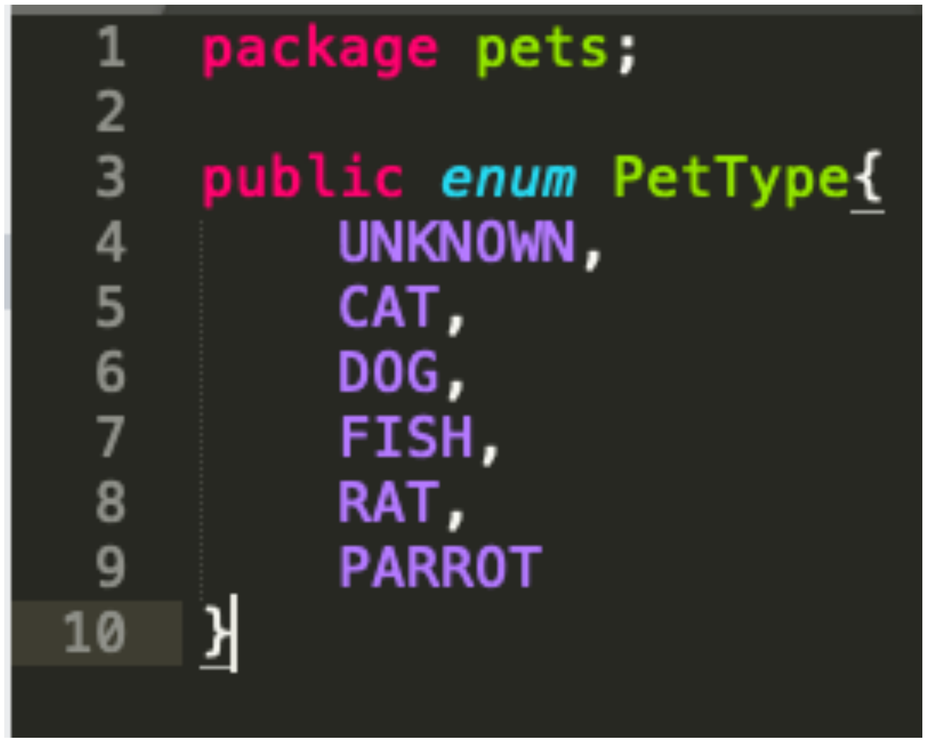
```
Root/pets/
    Pet.java // pet info
    PetType.java // enum
    program.java // main
```



pets/PetType.java

```
package pets;

public enum PetType{
        UNKNOWN,
        CAT,
        DOG,
        FISH,
        RAT,
        PARROT
}
```



pets/Pet.java

```java
package pets;

public class Pet{
        public String name;
        public PetType petType;
        public int age;

        public Pet()
        {
                name = "Unnamed";
                petType = PetType.UNKNOWN;
                age = 0;
        }

}
```

```java
package pets;

public class Pet{
    public String name;
    public PetType petType;
    public int age;

    public Pet()
    {
        name = "Unnamed";
        petType = PetType.UNKNOWN;
        age = 0;
    }

}
```

pets/program.java

```
package pets;

import pets.*;

public class program{
        public static void main(String[] args){

        Pet myPet = new Pet();

        System.out.println(myPet.name);

        }
}
```

```
1  package pets;
2
3  import pets.*;
4
5  public class program{
6      public static void main(String[] args){
7
8          Pet myPet = new Pet();
9          System.out.println(myPet.name);
10
11     }
12 }
```

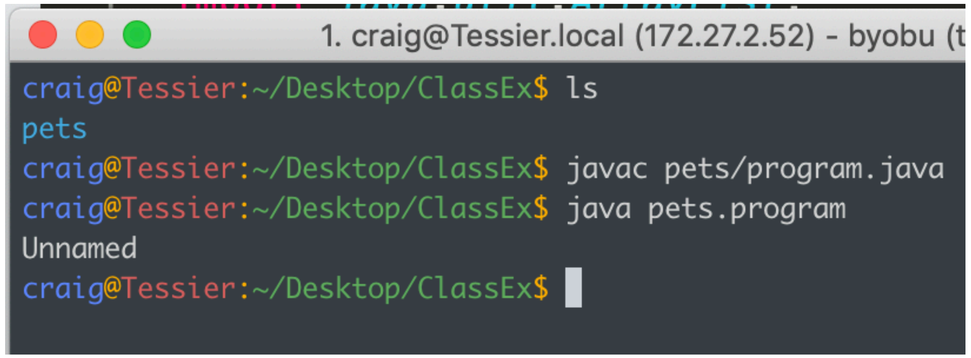
Compiling and Running

- Compile:

```
javac pets/program.java
```

- Run:

```
java pets.program
```



ClassPath (and others)

- The -cp argument specifies the path to classes (This is an override!)
- The -d argument specifies the path of the destination folder

Constructors

- Set the default properties of our object when we insatiate it with the new keyword

```
public Pet()
{
    name = "Unnamed";
    petType = PetType.UNKNOWN;
    age = 0;
}
```

Constructors can be overloaded

- Overloading:
  - Two or more functions with the same name
  - Have different signatures (arguments)

```
public Pet()
{
    name = "Unnamed";
    petType = PetType.UNKNOWN;
    age = 0;
}

public Pet(String name)
{
    this.name = name;
    petType = PetType.UNKNOWN;
    age = 0;
}
```

Noroff
School of technology
and digital media

Wednesday - Week 1

Tasks

Task 7: Git primer

- For those who have not had experience with git
  - Work through my GitIntro slides
  - Read the following tutorial: https://guides.github.com/activities/hello-world/

Task 8: Upgraded Name Search

- Modify your name search solution to have the following:
  - A separate class that describes a person
    - Reminder: It will need it's own source file
    - First Name, Last Name, Telephone, etc
    - Use overloaded constructors*
  - Create a collection of this new class
  - Modify your search to access a public method within this class

Task 9: File properties

Build a Java application that allows me to do the following:

Open: opens a specified file (the file will be a small .txt file)

Display statistics for the file:
-The name of the file that I chose.
- How big the file is.
- How many lines the file has.
- Let me search if a specific word exists in the file,
- How many time the specific word is found in the file.
- The search must ignore case.