

Objects in Java

Basic Concepts

- what is a class?
- defining classes
- creating objects

The Class definition is like a blueprint or design or specification



- here we specify the properties and actions of our class
- properties
 - name
 - breed
- actions
 - bark
 - wag tail

Create objects using class design

The tail shall be a discontinuous stream of the wind. It should be very thick at the base, gradually tapering toward the tip, of medium length, and extending no longer than to the back.

The ears shall be moderately sized for the head, set rather far back, and carried low on the skull, slightly above eye level.

The back shall be strong and the spine to extend from the withers to the croup, which extends to the rump.

The hindquarters shall be broad, muscular and well-developed. The hip on the back side well rounded and strong, short loins.

The hind legs shall be strongly bent, muscled with moderate angulation at the withers, and powerful, clearly defined thighs.

Feet are strong and compact, with well arched toes and a well-developed pad.

Head, strongly oval, tapering toward occiput, well defined and alertness on a level with the forehead. There should be no pendulous ears, set well apart, and neither pointing nor droop out.

The head shall be elegant and free from blemish, showing the bone structure of the skull, especially beneath the eye, in 3/4 view prominent in the neck.

The eyes shall be wide and be strongly well-developed.

The ears shall be powerful and free from substance, the inside surface being good, neither set close nor curly.

The neck shall be of proper length to allow the dog to extend its neck easily, without necessity. The neck should rise straight from the shoulders with a moderate arch.

The shoulder shall be distinguished from the rest of the spine, in a moderately well-sharp curve.

The shoulder shall be well laid back, long and sloping, joining on angle with the upper arm at approximately 90 degrees, the forearm the dog to allow the foreleg to set in a ready manner with strong forward reach.

Forequarters shall be muscular, well conditioned and balanced with the hindquarters.

Proportions shall be muscular, well conditioned and balanced with the hindquarters.

PERSONAL INFORMATION

NAME: _____

SEX: _____

BORN: DATE _____ LOCATION: _____

MOTHER: _____

FATHER: _____

FAC. DOBBY: _____

FAC. MUM: _____

WEIGHT: _____

COLOR: _____

ELEVATION
NOT TO SCALE

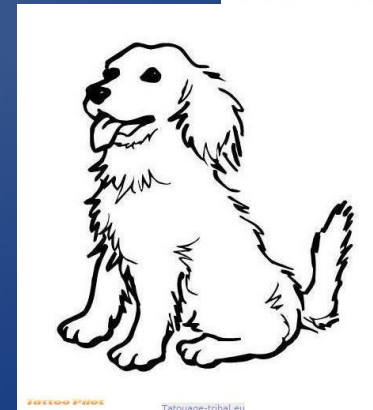
NAME OF DOG _____

BREED OF DOG _____

FRANK LLOYD WOOD
CANINE ARCHITECT

WET NOSE &
WIGGLY BUTTS, P.C.

GENERAL NOTES & DETAILS
D-1

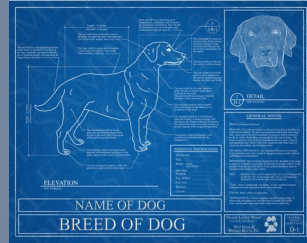


FRANK LLOYD WOOD
CANINE ARCHITECT

Tatouage-tribal.au

Create objects using class design

File: Dog.java



```
class Dog
{
    String name;
    int age;
    String breed;
    int tailPosition = 0;

    void bark()
    {
        println("woof!");
    }

    void wagTail()
    {
        tailPosition = -5;
        delay(1); // 1 second
        tailPosition = +5;
        delay(1); // 1 second
        tailPosition = 0;
    }
}
```

File: DogPark.java

```
public static void main(...)
{
    // create some dog objects
    Dog dog1 =
        new Dog("Fido", "Lab");

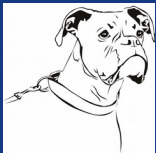
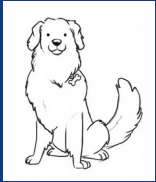
    Dog dog2 =
        new Dog("Rex", "Boxer");

    Dog dog3 =
        new Dog("Buddy", "Spaniel");

    dog1.bark();

    dog2.wagTail();

    dog3.bark();
    dog3.wagTail();
}
```



Defining a Class

- define a class in a separate file with the same name as the class
 - Person.java
- define fields to hold data, or properties, of the class
 - name, age

```
class Person
{
    String name;
    int age;
}
```

Using a Class to Create Objects

- a class is just an idea
- an object is that idea made into something "real"
- create and use objects in a separate file from the class
 - TestPerson.java
- the "new" keyword asks for space in memory for object

```
// a regular old variable  
int count = 0;
```

```
// more complex variables  
String msg = "Hello";
```

```
double[] grades =  
    new double[4];
```

```
// create a new person  
Person p1 = new Person();
```

Object Data Fields

- data fields contain the properties of individual objects
- each object will have its own copies of its own data
- data fields can store basic data types, arrays, or even other objects

```
// create a new person
Person p1 = new Person();
Person p2 = new Person();

p1.name = "Arthur Dent";
p1.age = 44;

p2.name = "Ford Prefect";
p2.age = 32;

println(p1.name);
// output is "Arthur Dent"

println(p2.age);
// output is 32
```

What is an Object

- the simplest version is like a complex variable
- holds multiple pieces of data
- data can be different types
 - int, double, boolean, char, String
- this type of data structure is sometimes called a record
- data for a Person might include:
 - name (String)
 - age (int)
 - married (boolean)

Source Code: Person Class

```
// file1: Person.java

class Person
{
    String name;
    int age;
    boolean isMarried;
}
```

```
// file2: TestPerson.java

class TestPerson
{
    public static void main(String[] args)
    {
        int x;
        double y;

        Person p = new Person();

        p.name = "Fred";
        p.age = 25;
        p.isMarried = false;
    }
}
```

Source Code: Person Class

```
// file1: Person.java

class Person
{
    String name;
    int age;
    boolean isMarried;
}
```

- the class (or record) is defined in its own file
- each part of the record is then defined within the class

Source Code: Person Class

- to use the record, make an object which has the properties of the class definition
- create a variable to identify the object
- use the "**new**" keyword to create the object in RAM

```
// file2: TestPerson.java

class TestPerson
{
    public static void main...
    {
        int x;
        double y;

        Person p = new Person();

        p.name = "Fred";
        p.age = 25;
        p.isMarried = false;
    }
}
```