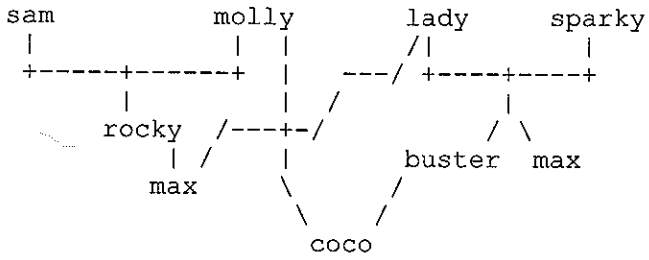
lecture 1

1f




```
class Dog {

        // 1a
        String name;
        boolean isMale;

        // 1d
        Dog mom, daddy;


        public Dog(String name, boolean isMale, Dog mom, Dog daddy) {
                // 3a
                this.name = name;
                this.isMale = isMale;

                // 3d (in addition to the two extra arguments to the constructor)
                this.mom = mom;
                this.daddy = daddy;
        }

        // 1b
        public String toString() {
                return name + " (" + (isMale ? "male" : "female") + ")";
        }

        public static void main(String[] args) {
                // 1c version
                // Dog refTomax         = new Dog("Max", true),
                // refTorocky   = new Dog("Rocky", true),
                // refTosparky  = new Dog("Sparky", true),
                // refTobuster  = new Dog("Buster", true),
                // refTosam     = new Dog("Sam", true),
                // refTolady    = new Dog("Lady", false),
                // refTomolly   = new Dog("molly",false),
                // refTococo    = new Dog("Max", false);

                // 1e version
                Dog sam         = new Dog("Sam", true, null, null),
                molly   = new Dog("Molly", false, null, null),
                rocky   = new Dog("Rocky", true, molly, sam),
                lady    = new Dog("Lady", false, null, null),
                sparky  = new Dog("Sparky", true, null, null),
                buster  = new Dog("Buster", true, lady, sparky),
                max     = new Dog("Max", true, lady, sparky),
                coco    = new Dog("Coco", false, molly, buster);

                // 1g
                System.out.println(coco.fathersName());
                System.out.println(sparky.fathersName());

                // 1h
                System.out.println(coco.hasSameMotherAs(rocky));
        }

        // 1g
        public String fathersName() {
```

```
                return daddy == null ? "Unknown" : daddy.name;
        }

        // 1h
        public boolean hasSameMotherAs(Dog otherDog) {
                // only if the input is valid
                // and only if this mom is equal to the other mom
                // and only if this mom is not unknown---otherwise two dogs with
                // unknown moms would return true..
                if(otherDog != null && otherDog.mom == this.mom && mom != null)
                        return true;
                return false;
        }
}
```

2a

```
class ITUStudent  {
        String fullName;
        int CPR;
        String BachelorDegree;
        int ITUGradeAverage;

        public boolean equals(Object other) {
                // See slide 6 in lecture 1.
                if (other == null) return false;
                if (other == this) return true;
                if ( !(other isinstanceof ITUStudent) ) return false;
                return CPR.equals(((ITUStudent) other).CPR);
        }
}
```

2b

```
class ITUStudent  {
        String fullName;
        int CPR;
        String BachelorDegree;
        int ITUGradeAverage;

        public boolean equals(Object other) {
                // See slide 6 in lecture 1.
                if (other == null) return false;
                if (other == this) return true;
                if ( !(other isinstanceof ITUStudent) ) return false;
                return ITUGradeAverage == other.ITUGradeAverage;
        }
}
```

3a

```
System.out.println( getName() + " " + getLitersPrHour())
```

3b

null, since uninitialized fields are set to null when objects are created

3c

'this.name' always refers to the variable in the object instance. 'name' refers to the argument variable in this case, as there is a name clash between the instance variable and the argument variable.

3d

```
public double totalLitersPrHour(){      // removed instance parameter and 'static' keyword,
        double total = source.getLitersPrHour();
        for (int index = 0; index < getNumberOfTributaries(); index++)
                // change from calling on that instance to just invoke the methods
                total = total + getTributary(index).totalLitersPrHour();
        return total;
```

3e
A this is needed to make it unambigous who the receiver of the "method call message" is.
Recall, 'this' is silently inserted by the java compiler if omitted.

3f
Static methods live outside any object instances, hence, the this reference is meaningles.

3g
```
Foo.java:3: non-static variable this cannot be referenced from a static context
      this.m();
      ^
```
and

```
Foo.java:3: non-static method m() cannot be referenced from a static context
       m();
       ^
```

for the program

```
class Foo {
        static void bar() {
                this.m();
                m();
        }

        public static void main(String[] a){
                Foo.bar();
        }

        public void m() { }
}
```

Exercise 1-1.h