ClassName.java

```
public class ClassName{
    //fields
    // constructors
    //methods
}
```

Tester.java

```
public class Tester{
    public static void main(String[] args){
        //client
    }
}
```

Fields
↳ instance fields: visibility type name;
→ class fields: visibility static type name;

Methods
↳ constructors: public ClassName(~)
→ instance method: public returnType functionName(~)
→ class method: public static returnType functionName(~)

this.

Override
↳ .equals - to compare objects
→ .toString() - to use print and println.

03 - Intro to Objects & Classes

01 - Creating Objects
↳ ClassName var = new ClassName();
→ Notes:
- accessing fields: objName.fieldName; (instance, public field)
- *Setting two objects equal merges them*

02 - Constructors
↳ You usually overload them — overwrite
→ public ObjName(~~) — override
public Fraction(int n, int d) — overload

02 - Instance Methods
public returnType name(~~);
↳ no static, has this
→ primitive types are passed in by value & copied.
→ objects as parameters are passed in by ~~reference~~.
→ In Client:
objName.functionName(~~);

05 - Class Methods
public static returnType name(~~);
↳ has static
→ In Client:
className.functionName(~~);

04 - Hiding Information
*↳ public
- fields are public by default

*→ private
- Limited to methods within the same class
- Accessor methods allow us to gain access to VALUES from private fields
- Mutator methods allow us to CHANGE values of private fields.

→ protected - subclasses also get access

04- Class Fields & Constants
↳ instance field: private type name
→ class field
- public static type name;
  public static final type NAME; //constant
- *There is only A SINGLE COPY OF THE CLASS FIELD regardless of how many objects are created*
- usage:
  ClassName.fieldName;

05- Displaying & Comparing Objects
↳ When comparing two objects using ==, their references are compared. Unless obj1 = obj2, obj1==obj2 will return false.
→ To compare values, override .equals()
→ To use print or println, toString() needs to be implemented.
→ print & println call the toString() to print objs if the f^n exists.
  ↳ otherwise, memory address is printed.

Overrides: .equals() & .toString()

overwrite - new val completely
override - does not modify original, just changes for class
overload - same name, diff parameters

[public/private] [(static)] [(final)] [type] [name]

public ClassName(~~) constructor
public returnType name(~~) instance
public static returnType name(~~) class