Common mistakes (jv-oop)

Please don't add redundant empty lines to your code.
We don't need them after class declaration or method signature.

- Bad example:

```
public class Main {

    public static void main(String[] args) {

        System.out.println("Hello world!");
    }
}
```

- Improved example:

```
public class Main {
    public static void main(String[] args) {
        System.out.println("Hello world!");
    }
}
```

Write informative messages in methods.

- Use English only and make messages informative
- The message should indicate what type of `Machine` is working right now `Truck`, `Bulldozer` or `Excavator`.

Use abstract references instead of specific ones where possible:

- Bad example:

```
Cat fluffy = new Cat();
Dog oscar = new Dog();
```

This example is bad cause it won't allow us to use polymorphism in our code. Our reference is now bonded to a specific implementation, but it is always better to depend on the abstraction. Let's see how we can improve it:

- Improved example:

```
Animal fluffy = new Cat();
Animal oscar = new Dog();
```

Depending on the case, class elements should have different access modifiers
Remember that if you don't use any access modifiers that will apply the default one. Do we always want to have all elements with default access modifiers? Remind yourself about the encapsulation principle and when private or public should be used.
Write informative messages when you commit code or open a PR.
Bad examples of commit/PR messages: `done` / `fixed` / `commit` / `solution` / `added homework` / `my solution` and other one-word, abstract or random messages.