JAVA stack and heap (variable life cycle)

1. Instance variables are variables declared **inside a class** but **outside any method**. Local variables are variables declared **inside a method** or **method parameters**.
2. All local variables live on the **stack**, in the frame corresponding to the method where the variables are declared.
3. Object reference variables work just like **primitive variables** – if the reference is declared as a local variable, it goes on the **stack**.
4. **All objects** live in the **heap**, regardless of whether the reference is a local or instance variable.
5. A local variable lives only **within the method** that declared the variable.
6. An instance variable lives as long as the object does. If the object is still alive, so are its instance variables.

	Life	Scope
Local Variables	A local variable is alive as long as its Stack frame is on the Stack.	A local variable is in scope only within the method in which the variable was declared.
Objects	An object is alive as long as there are live references to it.	You cannot use an object's remote control unless you've got a reference variable that's in scope.

7. Three ways to get rid of an object's reference:
   a. The reference goes out of scope.

```
public void go() {
      Life z = new Life();
}
```

   Reference z dies at end of method.

   b. The reference is assigned another object.

```
public void go() {
      Life z = new Life();
      z = new Life();
}
```

   The first object is abandoned.

   c. The reference is explicitly set to null.

```
public void go() {
      Life z = new Life();
      z = null;
}
```

   The first object is abandoned.