 In Java, all objects are allocated on the heap. There is no such thing as a stack variable. All variables are references by default. There is no concept of a pointer using a “\*” operator since everything is a pointer by default.

Student std;

However, declaring a variable does not create an object!

**In Java, no variable can ever hold an object.**

**A variable can only hold a reference to an object.**

std = new Student();

The expression greeting == "Hello" tests whether greeting

and "Hello" contain the same characters stored in the same memory location.

In Java, the destruction of objects takes place automatically.

If you create a class and don’t explicitly make it a subclass of

some other class, then it automatically becomes a subclass of the special class named Object.

(Object is the one class that is not a subclass of any other class.)

The practical effect of this in our example is that an object of type Car can be assigned to a variable of type Vehicle. That is, it would be legal to say Vehicle myVehicle = myCar; or even Vehicle myVehicle = new Car();

An instance method is always contained in an object. The class only contains the source code

for the method.

every class is a subclass of Object, a variable of type Object can refer to any object

 super.x always refers to an instance variable named x in the superclass.

The variable in the subclass does not replace the variable of the same name in the superclass; it merely hides it

When you write a method in a subclass that has the same signature as a method in its superclass, the method from the superclass is hidden in the same way. We say that the method in the subclass overrides the method from the superclass. Again, however, super can be used to access the method from the superclass.

Constructors are not inherited. That is, if you extend an existing class to make a subclass, the

constructors in the superclass do not become part of the subclass