A Java program is mostly a collection of objects talking to other objects by invoking each other's methods.

Every object is of a certain type, and that type is defined by a class or an interface.

**Final Classes**: When used in a class declaration, the final keyword means the class can't be subclassed. In other words, no other class can ever extend (inherit from) a final class, and any attempts to do so will give you a compiler error.

You should make a final class only if you need an absolute guarantee that none of the methods in that class will ever be overridden.

An **abstract class** can never be instantiated. Its sole purpose, mission in life, is to be extended (subclassed).

If even a single method is abstract, the whole class must be declared abstract.

When you create an **interface**, you're defining a contract for what a class can do,

without saying anything about how the class will do it. An interface is a contract.

Interfaces can be implemented by any class, from any inheritance tree.

Think of an interface as a 100-percent abstract class.

But while an abstract class can define both abstract and non-abstract methods, an interface can have only abstract methods.

One key rule for interface constants – They must always be “public static final”