Objects And Object Variables.

→ To work with objects, you first construct them and specify their initial state. then you apply methods to the objects.

→ Constructors always have the same name as the class name.

→ To construct a Date object, combine the constructor with the "new" operator.

→ new Date()

→ we can pass the object to a method

```
System.out.println(new Date());
```

→ we can apply a method to the object ~~after~~ ton that you just constructed.

Difference b/w objects and object variables.

```
Date deadline; // deadlines does'nt
                  refer to any object.
```

→ object variable, deadline, that can refer to objects of type Date.

→ The variable "deadline" is not an object and, in fact, does not even refer to any object yet. You cannot use any "Date" methods on this variable at this time.
We must first initialize the deadline variable.

☆ It is important to realize that an object variable does'nt actually contain an object. It only refers to an object.

→ In Java, the value of any object variable is a reference to an object that is stored elsewhere. The return value of the new operator is also reference.

```
Date deadline = new Date();
```

It has two parts "new Date()" makes an object of type "Date", and its value is a reference to that newly created object. The reference is then stored in the "deadline" variable.

☆☆ NOTE

→ People think that Java object variable behave like C++ reference. But in C++ there is no null reference, and reference cannot be assigned.

→ Think of Java object variables as analogous to object pointer in C++.

→ ~~Q~~ If we copy one variable to another, then both variables refer to the same ~~date~~ object, they are pointers to the same object. The equivalent of the Java null reference is the C++ NULL pointer.

☆ All Java objects live on the Heap. When an object contains another objects variable. It contain just a pointer to yet another heap object.

Mutator and Accessor Methods;

Simple Example

```
LocalDate aThousandDaysLater = newYearsEve.plusDays(1000);
```

→ The "plusDays" method yields a new "LocalDate" object, which is then assigned to the "aThousandDaysLater" variable.

→ the original object remain unchanged.

→ "plusDays" method does not mutate the object on which it is invoked

→ Methods that only access objects without modifying them are sometimes called accessor methods.

→ methods that change the state of objects after invoking it, are called mutator methods.