Recap:

1. Data types in Java:
   - Primitive data: int, double, char, boolean
   - Built-in objects: String, Scanner, char[] (or int[], boolean[]...), Random

2. Methods/Operations:
   - Primitive: +, -, *, <, ==
   - Built-in: equals(), nextInt(), [], nextInt()
   - Programmer-defined: replace, And, count()

3. Refresh:
   - Object: entity with attributes (data fields) and operations (methods).
   - Class: defines data fields & methods of an object.

4. Terms also apply to built-in objects.

   ex: String str = "hello";

   ↑ ↑ ↑
   class/ object/ value
   object variable (data-field)

Another data field? length
methods? Some: replace(), toUpperCase(), equals()
We will write our first programmer defined object to represent a BankAccount.

Note: 2 separate files/programs/classes.

First step: Think about which attributes/operations would be useful.

- Want to create an object for a Bank to store info about its customers.

attributes/data fields: account number (String), balance (double), name (String)

operations/methods (return type/parameter):
withdraw(), deposit(), checkBalance()
Simples approach first (not encapsulated)

→ save as BankAccount.java

public class BankAccount{

    String acctNum, name;
}

double balance;

// constructor - for creating a BankAccount object
public BankAccount(String num){
    acctNum = num;
    // typically an attribute that is unique to each object
    this.acctNum = num; // more explicit
}
public class BankDriver

public static void main()
{

BankAccount acct1 = new BankAccount("12345");

acct1acctNum = "12345"
	acctName = "Drumf"
	balance = 100

acct1.name = "Drumf"
acct1.balance = 100;

Back to BankAccount class

write the methods...

method header now:

public static return-type name (parameter_list)

public double getBalance()

return this.balance;

public void withdraw (double amt)

this.balance -= amt;
public void deposit(double amt) {
    this.balance += amt;
}

Back to BankDriver:

acct1.deposit(50);
acct1.withdraw(200);

balance = 1850 - 50

In-class code:

Suppose don't know acct # when we want to create objects:
can make an empty constructor:

// empty constructor
public BankAccount() {
}

Back in BankDriver:

BankAccount acct2 = new BankAccount();
acct2.acctNum = "S67889";
Back to BankAccount:

Another important method: toString()

//returns String of info about this object
public String toString()
{
    return this.acctNum + " " + this.acctName + " " +
    this.balance;
}

Back to Driver:

System.out.println(acct1);

(instead of: printIn(acct1.toString());)

Output: "12345" + Drumpf + " " = 50

Sometimes may want different versions of the same method
ex: withdraw() that always withdraws $20.

method overloading - using the same name for multiple
methods. (How will the compiler know which
method to call?) => parameter lists must be
different
BankAccount:

```java
public class BankAccount {

    ① final double AMOUNT = 20; // value can't be changed

    ② public void withdraw() {
        this.balance -= AMOUNT;
    }

    ③ AMOUNT = 30; // won't compile!

}
```

This was simplest approach, now more sophisticated.

Objects provide abstraction - we know they do something but we don't know how (implementation details).

Ex: Scanner: nextInt()  String: charAt()...

Only the object's class deals with implementation details.

Encapsulation - an object should be modifiable only by the object's methods.