Creating and Using Objects in Java and how to call their Methods

- The first thing we need to have to create an object is a class that defines what the object will look like. We’ll eventually learn to write our own classes, (!) but for now we’ll use classes that already exist.

- For this example, we will use a very simplified “Savings Account” class.

- We need to know what this class offers: what methods it has, and what information we can or should provide when we make objects of this class.

- There are a few ways we can get this information. One way is to look at the code of the class, but that includes a lot of details that we don’t care to know in order to use the class. One of the reasons we like using classes is that we don’t have to worry about all of the details in the class once its written.

- Another way we can look at the class is to look at the UML class diagram for the class. This gives us a nice overview of the class in a standard format.

Graphical user interface, text, application, email

Description automatically generated

- Here’s the class diagram for the “Savings Account” class. It shows us the data the objects contain though that’s not a major concern for us when using the class. However, it also shows us what methods the class provides. In this particular case, we have 3 constructors, which is how we will actually create the savings account class.

1. SavingsAccount()

Doesn’t take any information.

1. SavingsAccount(double startBalance)

Takes a start balance.

1. SavingsAccount(double startBalance, double interest)

Takes a start balance and also specifies the interest rate on an account.

- We also have 4 different methods:

1. double getBalance()

“Return the balance” Finds out what the balance is right now.

1. void deposit(double amountToDeposit)

This takes an amount to deposit and adds it to the balance.

1. void withdraw( double amountToWithdraw)

This takes an amount to withdraw and subtract from the balance.

1. processEndOfMonth()

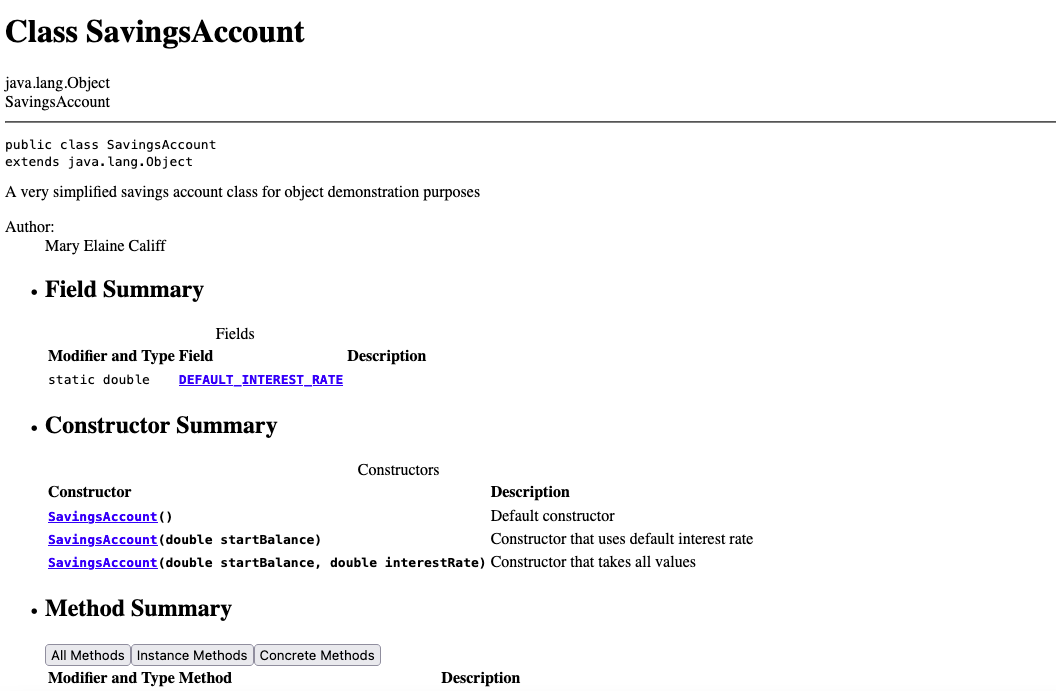
This handles end of month processing for the account.

- Looking at the UML class diagram is a useful tool, but it doesn’t actually provide any explanations. Therefore, we often look at API documentation for the class. Java provides a standard format for such documentation and a standard way to generate it, which is called Java Doc.

- We’ll look at the Java Doc for this savings account class, an .html file.

It will show us the name of the class and then we see summary information which gives us some of the information we saw on the class diagram so this is very similar except we have some descriptions as well as the actual information about calling the class and what the code looks like. In addition to this summary information that matches what’s in the class diagram, we do also have details.

- [( .html file discussed )](https://reggienet.illinoisstate.edu/access/lessonbuilder/item/8070265/group/97cf3c29-ad7e-4e75-8689-0e01d47cf51c/Source%20Code/SavingsAccount.html)



- There are cases where complex classes will not be self-explanatory, so having the Java Doc is useful when the class diagram is not sufficient.

- Before I create my class, I need to have access to the “SavingsAccount” class.

- To create an object, we are setting up a variable. The first thing we need for any variable is a declaration. And the type for an object will be the class that the object belongs to. In this case, it will be “SavingsAccount”.

public class ObjectPractice

{

public static void main (String[] args)

{

SavingsAccount acct1;

- Because I’ve imported the “SavingsAccount” class into the source (src), I can declare it as a data type and assign a variable to it (acct1) or (joesAcct).

- At this point I’ve declared my variable but just like primitive data types/ primitive variables, I haven’t assigned a value to it yet, so I can’t use it yet. (uninitialized)

- The way that I give an object variable a value is to use the keyword “new”. Then I need to call a constructer which tells Java what class I’m using and what information, if any I want to provide.

- After I type “new” and “SavingsAccount”, I type what is called the default constructor. It’s called default because it gets all default values that we did not provide. (We did not provide any special values for this specific object.

SavingsAccount acct1 = new SavingsAccount();

- Now we need to do things with the object. If we go back to the .html file, “SavingsAccount.html” , we can view the methods associated with the “SavingsAccount()” class.

- We’re going to find what the balance is in the account. This is the “getBalance” method which is a double data type. We’re going to use this in our code.

SavingsAccount acct1 = new SavingsAccount();

double accountBalance = acct1.getbalance();

- “accountBalance” is the name of the variable we’ve declared.

- “getBalance” is a method from the “SavingsAccount” class file that we can use to print the balance of an account, for this example, “acct1.”

What keyword do we use to create an object in Java?

1. create
2. construct
3. new
4. object

- The way calling methods works is we’re going to have the name of the object “the variable that refers to the object” and then a dot “.” And then the name of the method. Then sometimes we will need to provide some additional information.

- Which of the following tells us what options we have when creating an object?

A.

The constructors

B.

The list of methods

C.

The list of instance variables (also called attributes)

\*Check the file “ObjectPractice.java” in the IT168 folder for more commentary and the calling of the “SavingsAccount” class and it’s methods.