Question 1: Operators and Types

For each expression below, write the type of the expression on the first line, then the value of the expression on the line to the right. Write "*illegal*" if the expression is never allowed in java. If the expression might be allowed, assume it is used in a valid context. Only write "*illegal*" if there is absolutely no context where the expression is valid. Write "*unknown*" if the expression is legal but *impossible* to determine the value from what is given.

Expression	*Type*	*Value*	
$1 + 2$	int	3	$(*)$

$2 + \left(\frac{1}{4}\right)$	_____	_____	(1)
$(\text{double})2 + \frac{1}{4}$	_____	_____	(2)
$2 + \left(\frac{1.0}{4}\right)$	_____	_____	(3)
$\text{new Object}() == \text{new Object}()$	_____	_____	(4)
$\text{'b'} + (\text{char})0$	_____	_____	(5)
$x = (y == z)$	_____	_____	(6)
$2! = 3! = 4$	_____	_____	(7)
$\text{true} \mid\mid \text{false}$	_____	_____	(8)
$\text{false} == 1$	_____	_____	(9)
$(3 < 4)\ \&\&\ (5 < 6)$	_____	_____	(10)

Question 2: Strings

Write a method *capitalizeFirstK* that takes a String and an int value *k* as input and returns a String. The output should be the same as the input string except that the first *k* lower case letters are converted to upper case. You may use (but are not required to use) the following java API methods:

- boolean Character.isLetter(char ch)
- boolean Character.isLowerCase(char ch)
- char Character.toUpperCase(char ch)

```
> Midterm.capitalizeFirstK("How are you doing?", 4)
"HOW ARe you doing?"
```

Question 3: Short Descriptions

Please give precise answers to the following questions.

1. What is the purpose of a *type*?

2. What is *short-circuit evaluation* and what Java constructs use it?

3. What exactly happens if you do not provide a constructor for a class? Be precise.

Question 4a: Classes

Create a **BankAccount** class. The BankAccount should contain three values, an int *number*, a double *balance*, and a Customer *customer* (Assume Customer is a class that already exists).

- The **BankAccount** should be such that when we create an instance, the *number* must be included when we use new.
  (Example: new BankAccount(3345123))
- There should be a way to separately retrieve all three values and separately change the *balance* and *customer*

Question 5: Extending Classes

Suppose we have a class Customer that already exists.
The Customer class has the constructor:

- Customer(String name): creates a Customer instance with the given name and the method
- void deposit(double amount): deposits the given amount into the customer's account
  Create a class **BankCustomer** that extends Customer. The **BankCustoemr** class should have two attributes *checkingAccount* and *savingsAccount* both of type BankAccount of the previous question (4a).
- When an instance of BankAccount is created, the *name* must be included when we use new.
- Provide a means to retrieve and change the *checkingAccount* and *savingsAccount* values.
- When changing either the *savingsAccount* or the *checkingAccount*, that account's *customer* value should be set to this **BankCustomer** instance.

Question 6: Updating Inherited Methods

This question continues the class you started in Question 5.
The **BankCustomer** class inherits the deposit method of Customer.

- void deposit(double amount): deposits the given amount into the customer's account. Change how the deposit method works so that:
- If the *checkingAccount* exists, set the *checkingAccount*'s balance to be its existing balance plus the input amount.
- Otherwise, if the *savingsAccount* exists, set the *savingsAccount*'s balance to be its existing balance plus the input amount.
- Otherwise, call the inherited deposit methods with the given amount.