 

**Advanced Placement Computer Science**

**Unit 8: Polymorphism, Graphics, Inheritance**

**Lesson: Inheritance**

**FILLED OUT**

Inheritance is an amazing tool.

It allows one class to inherit the properties of another class

Consider the problem posed in Big Java on page 435.

A Bank offers two different types of accounts;

* A **checking** account has no interest, gives you a small number of free transactions per month and charges a transaction fee for each additional transaction
* A **savings** account earns interest that compounds monthly (we will compound using the balance on the **last** day of the month), no need to keep track of transactions

Here is a picture of the design from our text

Object

toString()

equals()

BankAccount

public BankAccount()

public BankAccount(double initalBalance)

public void deposit(double amount)

public void withdraw(double amount)

public double getBalance()

public void transfer(BankAccount other, double amount)

private double balance;

CheckingAccount

public CheckingAccount(double initialBalance)

public void deposit(double amount)

public void withdraw(double amount)

public void deductFees()

private int transactionCount;

private static final int FREE\_TRANSACTIONS = 3;

private static final double TRANSACTION\_FEE=2.0;

SavingsAccount

public SavingsAccount(double rate)

public void addInterest()

private double interestRate;

Some observations regarding java inheritance

All java classes extend Object implicitly

All methods except constructors are inherited

Objects are created from the inside out

Classes higher on the hierarchy are known asparent, super or base

Classes lower on the hierarchy are known as child, sub or derived

Children cannot directly access parents privates so they must also use accessor and mutator methods

Constructors in child classes can call parent constructors pass in essential parameters with the command super

MUST BE THE FIRST COMMAND INSIDE THE CONSTRUCTOR!!!!!!!!!!!!!!!!!!!!!!

Let’s look at the variables and methods in some objects that are created;

public class Tester{

 public static void main(String[] args) {

 BankAccount b1 = new BankAccount(5000);

 //You create a checking account with an initial balance of 2000 called c1

 CheckingAccount c1 = new CheckingAccount(2000);

 //You create a savings account with an initial balance of 1000 called s1

 SavingsAccount s1 = new SavingsAccount(1000);

}

//Lets draw the different “parts” of these objects

interestRate

withdraw() deposit()
balance

s1
toString(), equals()

transactioncount

withdraw() deposit()
balance

c1
toString(), equals()

withdraw() deposit()
balance

b1
toString(), equals()

A sub class usually follows the is a relationship

When extending an existing class, 3 possibilities exist for defining your methods

1. inherit: by default, you get all the goodies

2. add your own new methods.

3. over-ride or redefine existing methods (toString)

Only 2 possibilities exist for instance variables (also called fields )

1. inherit
2. add new variables (example turn variable from TicFrame)

If you have an over-ridden method, the way to call a parent method is super.methodName()

super.toString();

//Create an instance of the BankAccount class

BankAccount billG = new BankAccount();

//Deposit some money

billG.deposit(1e6);

//Withdraw some money

billG.withdraw(10);

//Get the balance

System.out.println(billG.getBalance());

//Create an instance of the Checking account with an initial deposit of $500

CheckingAccount ayoub$ = new CheckingAccount(500);

//Withdraw $ 3 times

ayoub$.withdraw(50);

ayoub$.withdraw(50);

ayoub$.withdraw(50);

//call the deduct fees method
ayoub$.deductFees();

//Get the balance

System.out.println(ayoub$.getBalance());