Assignment 2 – Bank Account

# Benjamin Matias | 991591184 | Trafalgar

public class Tester {

public static void main(String [] args){

//Savings Test //

// Savings(int account\_id, double balance, double overdraft) //

Account SVGTest = new Savings(12456, 156.00, 300.0);

System.out.println(SVGTest.toString());

SVGTest.deposit(200);

System.out.println(SVGTest.toString());

SVGTest.withdraw(600);

System.out.println(SVGTest.toString());

SVGTest.deposit(300);

System.out.println(SVGTest.toString());

// Chequing Test

Account CHQTest = new Chequing(55053, 500);

System.out.println(CHQTest.toString());

CHQTest.withdraw(100);

System.out.println(CHQTest.toString());

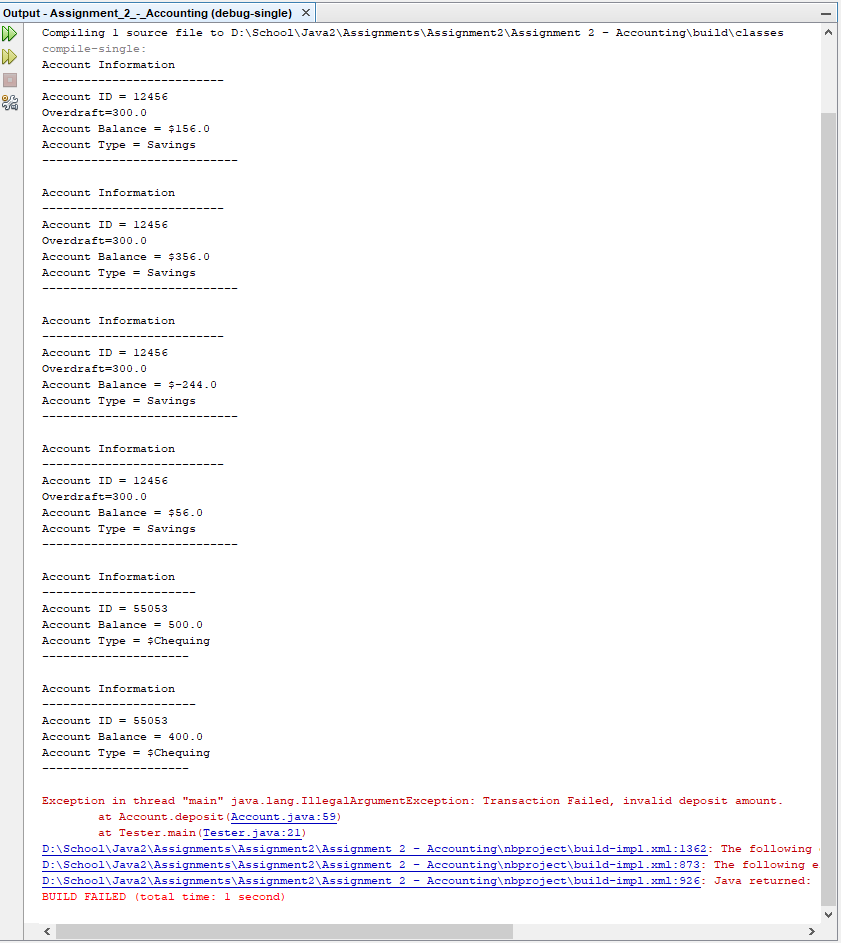
CHQTest.deposit(-50);

System.out.println(CHQTest.toString());

}

}

## Test Output



/\*

Author: Benjamin Matias

Program: Account.java

\*/

public abstract class Account {

public int account\_ID;

public double balance;

private String accountType;

Account() {

account\_ID = 0;

balance = 0;

accountType = "";

}

// Three Parameter

public Account(int account\_ID, String accountType, double balance) {

this.account\_ID = account\_ID;

this.accountType = accountType;

this.balance = balance;

}

// Getter and Setter for Account ID //

public int getAccount\_ID() {

return account\_ID;

}

public void setAccount\_ID(int account\_ID) {

this.account\_ID = account\_ID;

}

// Getter and Setter for Balance //

public double getBalance() {

return balance;

}

public void setBalance(double balance) {

if (balance >= 0) {

this.balance = balance;

} else {

throw new IllegalArgumentException("Invalid Balance Amount");

}

}

// Getter and Setter for Account Type //

public String getAccountType() {

return accountType;

}

public void setAccountType(String accountType) {

this.accountType = accountType;

}

// Deposit Method

public void deposit(double deposit\_amount) {

if (deposit\_amount >= 0) {

balance += deposit\_amount;

} else {

throw new IllegalArgumentException("Transaction Failed, invalid deposit amount.");

}

}

// Abstract method called Withdraw //

public abstract void withdraw(double withdraw\_amount);

}

/\*

Author: Benjamin Matias

Program: Savings.java

\*/

public class Savings extends Account {

private double overdraft, interest\_rate = 0, interest;

Savings() {

// Setting parent class values to empty values //

super(0, "", 0);

}

Savings(int account\_ID, double balance, double overdraft) {

// Initializing values for parent class //

super(account\_ID, "Savings", balance);

this.overdraft = overdraft;

}

public double getOverdraft() {

return overdraft;

}

public void setOverdraft(double overdraft) {

if (overdraft >= 0) {

this.overdraft = overdraft;

} else {

throw new IllegalArgumentException("Invalid Overdraft Amount");

}

}

public double getInterest\_rate() {

return interest\_rate;

}

public void setInterest\_rate(double interest\_rate) {

if (interest\_rate >= 0) {

this.interest\_rate = interest\_rate;

} else {

throw new IllegalArgumentException("Invalid Interest Rate");

}

}

public void addInterest() {

this.interest = this.balance \* this.interest\_rate;

}

public void withdraw(double withdraw\_amount) {

double withdraw\_limit = balance + overdraft;

if (withdraw\_amount <= withdraw\_limit && withdraw\_amount >= 0) {

balance -= withdraw\_amount;

} else if (withdraw\_amount > withdraw\_limit) {

throw new IllegalArgumentException("Transaction Failed, withdraw amount pass limit");

} else {

throw new IllegalArgumentException("Transaction Failed.");

}

}

public String toString() {

return "Account Information\n"

+ "--------------------------\n"

+ "Account ID = " + account\_ID

+ "\nOverdraft=" + overdraft

+ "\nAccount Balance = $" + balance

+ "\nAccount Type = " + getAccountType()

+ "\n----------------------------\n";

}

}

/\*

Author: Benjamin Matias

Program: Chequeing.java

\*/

public class Chequing extends Account {

private double transaction\_fee = 0;

Chequing() {

// super(int account\_ID, String accountType, double balance) //

super(0, "", 0);

}

// Two Parameter Constructor

Chequing(int account\_ID, double balance) {

super(account\_ID, "Chequing", balance);

}

// Chequing withdraw method //

public void withdraw(double withdraw\_amount) {

if (withdraw\_amount <= balance && withdraw\_amount > 0) {

balance -= (withdraw\_amount + transaction\_fee);

} else if (withdraw\_amount == 0) {

balance -= 0;

} else {

throw new IllegalArgumentException("Transaction Failed.");

}

}

// Outputs String

public String toString() {

return "Account Information\n"

+ "----------------------\n"

+ "Account ID = " + account\_ID

+ "\nAccount Balance = " + getBalance()

+ "\nAccount Type = $" + getAccountType()

+ "\n---------------------\n";

}

}