

More with Classes

Banking

Nov 27

Penn Percentage Project Survey

https://qfreeaccountssjc1.az1.qualtrics.com/jfe/form/SV_eJrorLiFIM9eaW

Charity5

```
public class Charity5 {
    private String name;
    private int donationTarget;
    private int donationsReceived;

    public Charity5(String nm, int dt, int dr) {
        this.name = nm;
        this.donationTarget = dt;
        this.donationsReceived = dr;
    }

    public String getName() {
        return name;
    }

    public int getDonationTarget() {
        return donationTarget;
    }

    public int getDonationsReceived() {
        return donationsReceived;
    }

    public int adjustDonationsReceived(int dr) {
        donationsReceived += dr;
        return donationsReceived;
    }

    public double percentageOfGoal() {
        return (double) (donationsReceived * 100) / donationTarget;
    }

    public String toString() {
        return name + " has a donation target of " + donationTarget + ". It has received " + donationsReceived
            + " which is " + percentageOfGoal() + " of its goal.";
    }
}
```

This class does NOT have a main function!!!

```
% javac Charity5.java
```

```
% java Charity5
```

Error: Main method not found in class Charity5,
please define the main method as:

```
public static void main(String[] args)
```

or a JavaFX application class must extend
javafx.application.Application

User5

```
public class User5 {  
    public static void main(String[] args) {  
        Charity5 charity = new Charity5("ICF", 100000, 5000);  
        System.out.println(charity.toString());  
    }  
}
```

- User5 does have a main, so it can be "run".
- More interesting, when I compile User5, Charity5 will also get compiled
 - javac looks at all the classes used by the thing you are compiling, and any need compiling, javac will do so
 - How does javac determine "need compiling"

Path / Classpath

- How does java / javac find classes
 - the "classpath" !
 - a variation on the Unix path! (UNIX, Unix or unix??)
 - according to Wikipedia "*Unix* was the original formatting, but the usage of *UNIX* remains widespread"
 - also UNIX is trademarked and according to the owners is an adjective
 - the Unix 'which' command
 - By default, classpath is
 - current directory
 - classes in Java distribution

Activity

- Create 2 instances of Charity5.
 - put them in two variables in a main function
- Determine which instance is (percentage-wise) further from its goal
- Use the toString method to print information about that charity

- Repeat, but this time make 5 instances of Charity5
 - put them into an array
- Find the instance that is the furthest from its goal
- Use the toString method to print information about that charity

Banks and Bank Accounts

- Bank
 - Data
 - accounts
 - ...
 - Activities
 - Create Account
 -
- Bank Account
 - Data
 - Account Number
 - ...
 - Activities
 - Deposit
 - ...

Bank Account

```
public class BankAccount {
    private final String accountNumber;
    private String name;
    private double balance;

    public BankAccount(String actNo, String name, double startDep) {
        this.accountNumber = actNo;
        this.name = name;
        this.balance = startDep;
    }

    public void changeName(String newName) {
        this.name = newName;
    }

    public double getBalance() {
        return balance;
    }

    public String getName() {
        return name;
    }

    public String toString() {
        return accountNumber + " " + name + " balance:" + balance;
    }
}
```

```
// alternately, it would make sense to return new balance
    public boolean deposit(double dep) {
        if (dep < 0) {
            System.out.println("Cannot make a negative deposit");
            return false;
        }
        if (dep > 10000) {
            System.out.println("No. Would have to report this to the Treasury");
            return false;
        }
        balance += dep;
        return true;
    }

    public boolean withdrawal(double withdrawal) {
        if (withdrawal < 0) {
            System.out.println("Cannot make a negative withdrawal");
            return false;
        }
        if ((balance - withdrawal) < 0) {
            System.out.println("Not enough money");
            return false;
        }
        balance -= withdrawal;
        return true;
    }
}
```

Bank

```
public class Bank {
    BankAccount[] accounts = null;
    int accountNumber = 2;
    int activeAccounts;

    public Bank(int size) {
        accounts = new BankAccount[size];
        activeAccounts = 0;
    }

    private String nextAccountNumber() {
        accountNumber *= 1.5;
        String nextNum = "" + accountNumber;
        while (nextNum.length() < 9) {
            nextNum = " " + nextNum;
        }
        return nextNum;
    }

    public BankAccount makeAccount(String name, double initialDeposit) {
        BankAccount ba = new BankAccount(name, nextAccountNumber(), initialDeposit);
        accounts[activeAccounts] = ba;
        activeAccounts++;
        return ba;
    }

    public void listAccounts() {
        for (int i = 0; i < activeAccounts; i++) {
            System.out.println(accounts[i]);
        }
    }
}
```

User

- make 3 accounts, each with a starting balance of \$100
- 10 times
 - randomly pick an account
 - randomly pick an account in \$10-20
 - randomly pick deposit/withdrawal
- Print the accounts