Making Classes
When you make a class
(with more than just static methods)

- What data will it store?
  - How does the data get in?
- What updates to the data are allowed? How?
- What kind of data does it provide?
- Does it (should it) have a "printable" representation?
The String class

• What data will it store?
  • A string

• How does the data get in?
  • new String("string");

• What updates to the data are allowed? How?
  • None

• What kind of data does it provide?
  • indexOf, substring, startsWith, length, ....

• Does it (should it) have a "printable" representation?
  • Yes, just the string itself
Task: A Charity tracker

- What data will it store?
  - charity name, donations received, goal, ...
- How does the data get in?
- What updates to the data are allowed? How?
  - Just the donations received
- What kind of data does it provide?
  - donations received, percentage of goal, goal, ...
- Does it (should it) have a "printable" representation?
  - Yes, please
public class Charity1 {
    public String name;
    public int donationTarget;
    public int donationsReceived;
}

public class User1 {
    public static void main(String[] args) {
        Charity1 charity = new Charity1();
        System.out.println(charity.donationTarget);
        System.out.println(charity.donationsReceived);
        System.out.println(charity.name);
        charity.name = "International Crane Foundation";
        charity.donationsReceived = 10000;
        charity.donationTarget = 30000;
    }
}

Observations -- Yea
• it works -- ish
• The data for a charity is grouped into one place

Observations -- Boo
• cumbersome to set values of vars
• No controls on var values
• All the work needs to be done by user!

2 classes in 2 files
javac on User1.java will also compile charity1.java

public, not static, not a method "instance variables"

create an instance

instance variables all get default values -- like arrays
get to instance variables via "."

Set values of instance variables via "." also
Cumbersome to set

Set initial values using a "constructor"
- a special method
  - Purpose is to set the values of instance variables
  - Name of method must be the name of the class
  - NO return type -- it returns an instance of the class
  - Often does nothing but set values of instance variables
  - May be overloaded

```
public class Charity2 {
    public String name;
    public int donationTarget;
    public int donationsReceived;

    public Charity2(String nm, int dt, int dr) {
        this.name = nm;
        this.donationTarget = dt;
        this.donationsReceived = dr;
    }
}
```

"this." refers to the instance itself.
May be put before instance variables
 Usually optional, never wrong.

"super();" implicitly the first line
super() is what makes the space

Can add controls if needed
E.g., target > 0
name != null

Here is the constructor
If I do not write one, java supplies a "no parameter" constructor that just makes that space for the instance
public class User2 {
    public static void main(String[] args) {
        Charity2 charity = new Charity2("ICF", 10000, 20000);
        System.out.println(charity.donationTarget);
        System.out.println(charity.donationsReceived);
        System.out.println(charity.name);

        charity.name = "International Crane Fouundation";
        charity.donationsReceived = 10000;
        charity.donationTarget = 30000;
    }
}

Everything other than constructor is the same

Do we want to allow anyone to change these in any way?
Charity3

going private

- What data will it store?
  - charity name, donations received, goal, ...
- How does the data get in?
- What updates to the data are allowed? How?
  - Just the donations received
- What kind of data does it provide?
  - donations received, percentage of goal, goal, ...
- Does it (should it) have a "printable" representation?
  - Yes, please

```
public class Charity3 {
    private String name;
    private int donationTarget;
    private int donationsReceived;

    public Charity3(String nm, int dt, int dr) {
        this.name = nm;
        this.donationTarget = dt;
        this.donationsReceived = dr;
    }
}
```

"Best practice", but not required, is to make all instance variables "private".
Charity

getters and setters

• When instance vars are private, no one can see/use them directly
  • Sometimes this is good and intentional
    • FileReader -- where you are in the file
  • Sometimes just awkward

• Solution public get and set accessors
  • only write for those instances variables that you want to allow access
    • For Charity class, what should those be?
public String getName() {
    return name;
}

public int getDonationTarget() {
    return donationTarget;
}

public int getDonationsReceived() {
    return donationsReceived;
}

public void setDonationsReceived(int dr) {
    donationsReceived = dr;
}
Charity4
Providing the right data

• What data will it store?
  • charity name, donations received, goal, ...
• How does the data get in?
• What updates to the data are allowed? How?
  • Just the donations received
• What kind of data does it provide?
  • donations received, percentage of goal, goal, ...
• Does it (should it) have a "printable" representation?
  • Yes, please

public class Charity4 {
    private String name;
    private int donationTarget;
    private int donationsReceived;

    public Charity4(String nm, int dt, int dr) {
        this.name = nm;
        this.donationTarget = dt;
        this.donationsReceived = dr;
    }

    // getters not shown

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

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

A transformation of the data. In this case, uses only instance variables
Charity4 toString

• What data will it store?
  • charity name, donations received, goal, ...
• How does the data get in?
• What updates to the data are allowed? How?
  • Just the donations received
• What kind of data does it provide?
  • donations received, percentage of goal, goal, ...
• Does it (should it) have a "printable" representation?
  • Yes, please

```java
public class User4 {
    public static void main(String[] args) {
        Charity4 charity = new Charity4("ICF", 10000, 20000);
        System.out.println(charity);
    }
}
```

% javac User4.java
% java User4
Charity4@1eb44e46

UGH .. This looks a lot like print representation of an array!
toString()
Every class has one

• Default toString method is ugly
  • Charity4@1eb44e46

• So "override" with one of your own

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

    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."
    }
}
Activity

• Create 2 instances of Charity5.
  • put them in two variables in a main function
• Determine which instance is (percentagewise) 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