Announcements

- We're pleased with how many people are already working on A1, as evidenced by Piazza activity.
- Please be sure to look at Piazza note @54 every day for frequently asked questions and answers.
- Groups: Forming a group of two? Do it well before you submit – at least one day before. Both members must act: one invites, the other accepts. Thereafter, only one member has to submit the files.
- A2: Practice with strings
  - We will give you our test cases soon!

References to text and JavaSummary.pptx

- A bit about testing and test cases
- Class Object, superest class of them all.
- Text: C.23 slide 30
- Function toString() C.24 slide 31-33
- Overriding a method C.15–C.16 slide 31-32
- Static components (methods and fields) B.27 slide 21, 45
- Java application a program with a class that declares a method with this signature:
  public static void main(String[])

Homework

1. Read the text, about applications: Appendix A.1–A.3
2. Read the text, about the if-statement: A.38–A.40
3. Visit course website, click on Resources and then on Code Style Guidelines, Study
   2. Format Conventions
   4.5 About then-part and else-part of if-statement

A bit about testing

Test case: Set of input values, together with the expected output.

Develop test cases for a method from its specification --- even before you write the method’s body.

/** return the number of vowels in word w. */
/** Precondition: w contains at least one letter and nothing but letters */
public int numberOfVowels(String w) {
  ... } 

How many vowels in each of these words?
creek
syzzygy
yellow

Developing test cases first, in “critique” mode, can prevent wasted work and errors

Class W (for Worker)

/** Constructor: worker with last name n, SSN s, boss b (null if none). */
public W(String n, int s, W b)
/** = worker's last name */
public String getName(){}
/** = last 4 SSN digits */
public String getSsn(){}
/** = worker's boss (null if none) */
public W getBoss(){}
/** Set boss to b */
public void setBoss(W b)

W of
name Obama
ssn 23456789
boss null

W(…) getName() getSsn() getBoss() setBoss(W) toString() equals(Object) hashCode()
Class Object: the superest class of them all

Java Every class that does not extend another extends class Object. That is, public class W {...} is equivalent to public class W extends Object {...}

We draw object like this

We often omit this partition to reduce clutter; we know that it is always there.

A note on design

Don't use `extends` just to get access to hidden members!

A should extend B if and only if A “is a” B

A PhD tester is not a PhD Student!

An elephant is an animal, so Elephant extends Animal

A car is a vehicle, so Car extends Vehicle

An instance of any class is an object, so AnyClass extends java.lang.Object

The inheritance hierarchy should reflect modeling semantics, not implementation shortcuts

What is “the name of” the object?

The name of the object below is PhD@aa11bb24

It contains a pointer to the object –i.e. its address in memory, and you can call it a pointer if you wish. But it contains more than that.

Variable c, declared as PhD c, contains not the object but the name of the object (or a pointer to the object).

Method toString

toString() in Object returns the name of the object: W@af

In appropriate places, the expression c.toString() calls this method

Another example of toString()

An instance represents a point (x, y) in the plane */

public class Point {
    private int x; // x-coordinate
    private int y; // y-coordinate
    ...
    /** repr. of this point in form ”(x, y)” */
    public String toString() {
        return "c.toString() calls this method";
    }
}

Function toString should give the values in the fields in a format that makes sense for the class.
What about this

- **this** keyword: this evaluates to the name of the object in which it occurs
- Makes it possible for an object to access its own name (or pointer)
- Example: Referencing a shadowed class field

```java
class Point {
    public int x = 0;
    public int y = 0;
    //constructor
    public Point(int x, int y) {
        x = x;
        y = y;
    }
}
```

Inside-out rule shows that field x is inaccessible!

Intro to static components

```java
/** = "this object is c's boss". 
   Pre: c is not null. */
public static boolean isBoss(W b, W c) {
    return b == c.getBoss();
}
```

**static** there is only one copy of the method. It is not in each object

Good example of static methods

- **java.lang.Math**

  [Link](http://docs.oracle.com/javase/8/docs/api/java/lang/Math.html)

Java application

```java
public static void main(String[] args) {
    // ...
}
```

Running the application effectively calls method `main`

Command line arguments can be entered with `args`
Use of static variables: Maintain info about created objects

```java
public class W {
    private static int numObs; // number of W objects created
    /* Constructor: */
    public W(…) {
        …
        numObs = numObs + 1;
    }
}

To have numObs contain the number of objects of class W that have been created, simply increment it in constructors.
```

Uses of static variables: Implement the Singleton pattern

```java
public class Singleton {
    private static final Singleton instance = new Singleton();
    private Singleton() { }
    public static Singleton getInstance() {
        return instance;
    }
    // … methods
}
```

Class java.awt.Color uses static variables

```java
public static final Color black = …;
public static final Color blue = …;
public static final Color cyan = new Color(0, 255, 255);
public static final Color darkGray = …;
public static final Color gray = …;
public static final Color green = …;
…
```

An instance of class Color describes a color in the RGB (Red-Green-Blue) color space. The class contains about 20 static variables, each of which is (i.e. contains a pointer to) a non-changeable Color object for a given color.