Announcements

- A0 has been graded
  - Everyone who submitted gets a grade of 1 (the max)
  - We’re not checking submissions! We wanted you to learn how to make sure that assert statements are executed.
- 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 @84 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 C15–C16 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, 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.

```java
/** = 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
- synyny

Class W (for Worker)

```java
/** Constructor: worker with last name n, SSN s, boss b (null if none).
  * Prec: n not null, s in 0..999999999 with no leading zeros.*/
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)
```

Contains other methods!
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 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
  - 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
- A PhDTester is not a PhD student!
- The inheritance hierarchy should reflect modeling semantics, not implementational 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 e, declared as PhD e, contains not the object but the name of the object (or a pointer to the object).

c toString() calls this method

c toString() calls this method

Method toString

toString() in Object returns the name of the object: W@af
Java Convention: Define toString() in any class to return a representation of an object, giving info about the values in its fields.
New definitions of toString() override the definition in Object.toString()
In appropriate places, the expression c automatically does c.toString()

toString() calls this method

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 "(" + x + ", " + y + "\)";
  }
}

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
public 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
   public static boolean isBoss(W b, W c) {
       return b == c.getBoss();
   }
   ```

Intro to static components

- ```java
   public boolean isBoss(W c) {
       return this == c.boss;
   }
   ```

```
\*\* = "this object is c's boss".
Pre: c is not null. */
```

- Spec: return the value of this true-false sentence. True if this object is c's boss, false otherwise.

```java
x.isBoss(y) is false
y.isBoss(x) is true
```

```
\*\* = "b is c's boss".
Pre: b and c are not null. */
```

```
Preferred:
```

```java
isBoss(x, y)
```

```
Box for W (objects, static components)
```

Good example of static methods

- java.lang.Math

```
http://docs.oracle.com/javase/8/docs/api/java/lang/Math.html
```

Java application

- Java application: bunch of classes with at least one class that has this procedure:

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

  Type String[]): array of elements of type String. We will discuss later.

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.

Class `java.awt.Color` uses static variables

```java
public static Color black = ...;
public static Color blue = ...;
public static Color cyan = ...;
public static Color darkGray = ...;
public static Color gray = ...;
public static Color green = ...;
```

Uses of static variables: Implement the Singleton pattern

```java
public class Singleton {
    private static final Singleton instance = new Singleton();

    private Singleton() {
        // ... constructor
    }

    public static Singleton getInstance() {
        return INSTANCE;
    }

    // ... methods
}
```

Only one Singleton can ever exist.