Java, design, and lessons from lab
Reminders

• Reading
  – For today: Introduction and Prelude
  – For Friday: Chapter 1

• Homework
  – Send me your Clicker ID
  – Codingbat problems, some due today and some Friday
  – ~1 page on your interest in computing/computer science (due Friday)

• If you borrowed a clicker on Monday (or today), send me an email to let me know
New announcement

• Course TAs:
  – Caleb Gumanow
  – Nicolette Laird
  – Dustin Loch
  – Rosemary Momoh
  – Alex Reddington

They’ll be helping in lab. Some will staff office hours (TBA).
Java review
Which of the following declares an array of integers?

A. `int[] nums;`

B. `integer[] nums;`

C. `nums = new int[10];`

D. `nums = new integer[10];`

E. Not exactly one of the above
Which of the following declares an array of integers?

A. `int[] nums;`
B. `integer[] nums;`
C. `nums = new int[10];`
D. `nums = new integer[10];`
E. Not exactly one of the above
Which of the following is the signature for a method that takes a Person and returns a Dwelling?

A. public Dwelling(Person resident)
B. public Dwelling getResidence(Person p)
C. public Person(Dwelling d)
D. public Person getResident(Dwelling d)
E. Not exactly one of the above
Which of the following is the signature for a method that takes a Person and returns a Dwelling?

A. `public Dwelling(Person resident)`
B. `public Dwelling getResidence(Person p)`
C. `public Person(Dwelling d)`
D. `public Person getResident(Dwelling d)`
E. Not exactly one of the above
Which of the following correctly replaces the contents of `nums` with all 0s?

A. ```
for(int i=0; i<10; i++)
    nums[i] = 0;
```  
B. ```
for(int i=0; i<nums.size; i++)
    nums[i] = 0;
```  
C. ```
for(int i=0; i<nums.length(); i++)
    nums[i] = 0;
```  
D. ```
int i = 0;
    while(i < nums.length)
        nums[i] = 0;
```  
E. Not exactly one of the above
Which of the following correctly replaces the contents of nums with all 0s?

A. `for(int i=0; i<10; i++)`
   `nums[i] = 0;`
B. `for(int i=0; i<nums.size(); i++)`
   `nums[i] = 0;`
C. `for(int i=0; i<nums.length(); i++)`
   `nums[i] = 0;`
D. `int i = 0;`
   `while(i < nums.length)`
   `nums[i] = 0;`
E. Not exactly one of the above (none are correct)
Which of the following returns whether integer x is between 5 and 10 (inclusive)?

A. if((x <= 5) || (x >= 10))
   
   return true;

B. if((x >= 5) && (x <= 10))
   
   return true;

C. return ((x >= 5) && (x <= 10));

D. return !((x < 5) || (x > 10));

E. Not exactly one of the above
Which of the following returns whether integer \( x \) is between 5 and 10 (inclusive)?

A. `if((x <= 5) || (x >= 10))`
   
   return true;

B. `if((x >= 5) && (x <= 10))`
   
   return true;

C. `return ((x >= 5) && (x <= 10));`

D. `return !((x < 5) || (x > 10));`

E. Not exactly one of the above (C & D)
Moving on...
Abstract data types (ADTs)

• Set of operations that are supported on a particular kind of data (like a Bag)

• Does not include implementation details
  – Language independent
  – May have different implementations that optimize different operations
Which of the following is not part of the Bag ADT?

A. An attribute num storing the number of values in the Bag
B. A size method that returns the number of values in the Bag
C. The size of the array storing values that are put into the Bag
D. The main function that creates a Bag and tests it by calling the methods
E. Not exactly one of the above
Which of the following is not part of the Bag ADT?

A. An attribute num storing the number of values in the Bag (no: implementation detail)
B. A size method that returns the number of values in the Bag
C. The size of the array storing values that are put into the Bag (only if there's maximum size)
D. The main function that creates a Bag and tests it by calling the methods (no)
E. Not exactly one of the above
Another example: Counter ADT

• Used to represent an integer value that will be increased over time (e.g. a game score)

• Operations:
  – create: makes a Counter with value 0
  – increment: increases the value
  – tally: returns the current value
Design principles

• Encapsulation / information hiding: Hiding unnecessary details from other parts of the program

• Abstraction: Focus on what the class does instead of how (methods provided rather than their implementation)
Object-oriented (OO) programming

- Program is bunch of interacting objects
- Classes (types of objects) specify what a kind of data an object stores (attributes) and what actions it can perform (methods)
- Objects are instances of a class with a specific value for each attribute
Java interfaces

- Essentially a list of method signatures

```java
public interface Measurable {
    public double getPerimeter();
    ...
}
```

Looks like a method except it has a semicolon instead of a method body
Using an interface

public class Circle implements Measurable {
    //must have methods for all promised signatures
}
public class Circle implements Measurable {
    //must have methods for all promised signatures
}

Allows variable to store any implementing type:

public void wrap(Measurable m) {
    //m can be from any class implementing Measurable
    //can only use properties of m in Measurable
}
Application Programming Interface (API)

• Set of calls provided to the programmer by an implemented library
• Roughly the implementation of an ADT

• Example: Google Maps provides an API so that programmers can display their own data on a map
Design ideas

• Use cases: develop “stories” of how the code will be used

• Figure out the classes needed (nouns in stories)

• Draw relationships between classes in Unified Modeling Language (UML)