Objects in Programming

Final Test Review

Outline

- Format
- Topics

Format – Taking the test

- 2 hours
- At the MECC
- Open book
- Bring whatever notes/books you would like
  - Subject to approval by the staff at the test site
  - They are a separate organization that has its own rules
- No electronics allowed (laptops, phone, etc.)
- The test itself is taken on a computer, which will be provided
  - There will be no IDE, compiler or network access

Format – Question types

- True/False
- Given some code, find the errors
- Given some code, what does this code print?
- Given a problem description, write code that solves the problem

Format – Question types

- Find errors
  - By line number
  - One error per line
  - Compile-time errors are allowed
  - Checked exceptions can occur
  - No errors depend on the value of a variable or input values
- Types of errors
  - Syntax (missing braces, parentheses, semi-colons, etc.)
  - Other (attempting to access a private field outside an object, accessing an instance field from a static method, no permissions to write to a file, etc.)

Format – Question types

- What does this code print?
  - Code is correct – it does not throw an exception or crash
  - For all code given, you can assume that all the correct imports are used
    - They are omitted in the test to save space
  - It is possible that restrictions are placed on the code
    - For example, it may be stated that the code does/does not have permissions to open a file
    - Or that a certain input variable is positive, or not a number, or that a file is empty, etc.
  - Try to be as close as reasonably possible to exactly what would be printed
    - Avoid extra commas, dashes, etc.

Format – Question types

- Writing a GUI
  - Expect about a page or so of code
  - You can ignore import clauses
    - We will assume they are there
  - Try to avoid major syntax errors
  - Swing and AWT

Topics – Methods

- Signatures
  - Know what makes up a method's signature
  - Be able to overload and override methods
  - Given two or more method definitions, know which one is called in a specific case
- Parameters – know the difference between objects and primitive types
  - Primitive types (int, float, char, etc.) are passed by copy and any changes do not stick after the method ends
  - Objects are passed by reference and any changes made by calling methods on the object do stick after the e=method ends

Topics – Classes and Interfaces

- Classes
  - Have constructors
  - Can have private, protected and default fields and methods
  - Can use new to create objects
- Interfaces
  - No constructors
  - Only have public methods and static fields
  - Cannot create new objects
  - But can declare variables and parameters of interface type

Topics – Classes and Interfaces

- Abstract classes
  - Know the difference between an abstract class and a non-abstract (concrete) class
  - Be able to declare an abstract class
  - Be able to inherit from an abstract class and create a concrete one

Topics – Inheritance

- Inheritance
  - Singly-rooted inheritance tree
  - Object at top
- Descendants and ancestors
  - Immediate and remote
- How to extend a class
  - Syntax
  - Single inheritance of classes
  - Only classes can extend classes

Topics – Inheritance

- How to implement an interface
  - Syntax
  - Multiple inheritance of interfaces
  - Both classes and interfaces can implement interfaces
- Know how to make fields available/hidden from subclasses
- Overloading vs overriding
  - The difference
  - How to do each
  - Effect of private keyword

Topics – Inheritance and Constructors

- Constructor chaining
  - Default behavior
  - Calling the parent class constructor using `super()`
  - Calling `super` with arguments
  - Calling the parent constructor first
- The default constructor
  - What is it?
  - How do you get one?
  - What does it do?
  - Relationship to no-argument constructor

Topics – Inheritance and Polymorphism

- Relationship
  - If class C extends class P, then an object of class C can be used whenever an object of class P could be used
- Effect on
  - Which methods can be called from class C
  - Which fields are visible both within and outside of class C
- Use in
  - Parameters
  - Variable declaration

Topics – Access Control

- Access Control Modifiers
  - public, private, protected and default
  - Know what effect these have on when fields and methods can be seen/used
    - Both in general
    - And in terms of inheritance/polymorphism
- The Encapsulation Principle
  - Make all fields private and add public setters and getters
  - The difference between the interface (what is available outside the class) and the implementation (how the work is actually done, usually private to the class)

Topics – Static and Non-static

- Static fields
  - Where they can be accessed from
  - A static field has the same value over all objects of the same class
  - Static fields are accessed through the class
- Non-static fields (aka instance fields)
  - Where they can be accessed from
  - An instance field can have different values over objects of the same class
  - Instance fields are accessed through objects

Topics – Static and Non-static

- Static methods
  - Where they can be accessed from
  - Static methods are accessed through the class
  - Static methods cannot access instance variables (except through objects)
- Non-static methods (aka instance methods or just methods)
  - Where they can be accessed from
  - Instance methods are accessed through objects
  - Instance methods can access static variables

Topics – GUI

- No JavaFX
- Swing and AWT
- Be able to
  - Open a window
  - Add a panel, buttons, text field, dropdowns, etc.
  - Handle any events
  - Draw geometric objects (rectangles, circles, lines, etc.)

Topics – Exceptions

- Given a method that throws an exception, be able to write a try-catch block that catches that exception
- Define an exception class
- Explicitly throw an exception
- Define a method that throws a checked exception
- Know the difference between
  - Errors
  - Unchecked exceptions
  - Checked exceptions

Topics – I/O

- Files
  - Open a file
  - Read character data out of it
  - Write character data to it
  - Close it
- Character data includes primitive types (int, float, char) and Strings