Northeastern University

Lecture 14: Mid-term Exam Review

Prof. Chen-Hsiang (Jones) Yu, Ph.D.

College of Engineering

Format

- The exam will be 5 - 6 problems, with some problems having multiple sub-questions.
- It is a closed-book exam. No calculators, books, phones, or anything else. You can only use a browser to take the exam. No extra tabs, no ChatGPT.
- To make exam consistent to all students, we will have online lectures on Tuesday, Feb. 28. The zoom link of the lecture is the same as the 2nd lecture. (Please reference to the syllabus.)

Format (cont.)

- It is an online exam, and it can be only available during the lecture time on Tuesday, Feb. 28.
- The link of the exam will be announced during the lecture time.
- During the exam, please make sure both of your camera and microphone are “on” during the whole exam time.

Content

- Kinds of questions to expect:
  - » Explain object-oriented concepts and terminologies
  - » Explain a program or part of a program
  - » Write your own code
  - » Fix incorrect code / find bugs in code
  - » Fill in the blank (in a program)
  - » Short answer
  - » Multiple choices questions

Content (cont.)

- Everything we've covered so far, including:
  - Mathematical expressions
  - `if-else` statements
  - `while`, `do-while`, and `for` loops
  - Methods
  - Array and ArrayList
  - File I/O and exceptions
  - Classes and Objects
  - Inheritance and Polymorphism

Review Exercises

- The following slides contain exercises that will help you prepare for the exam.
- The exercises give you an idea of the style of questions to expect as well as the complexity.

Exercise

- What is static variable?
- What’s the difference between static variables and instance variables?

Answer

- Static variables:
  - » The variables that can be shared by all objects of a class
  - » Static variables also called *class variables*
- Instance variable:
  - » The variables that belong to an object
  - » Both static variables and instance variables are sometimes called *fields* or *data members*
- Differences:
  - » Each object has one copy of its instance variables. They are not shared with other objects with the same class type.

Answer

Member Variables

Instance Variable

```
class City
{
   int count;
   ...
}
```

Class Variable

```
class City
{
   static int count;
   ...
}
```

Exercise

- What is the meaning of Inheritance? Please give an example of it.

Answer

- Inheritance allows you to define a very general class and then later define more specialized classes that add some new details to the existing class definition.
- Example:

```
public class Person{
    private String name;
    public Person(){
        …
    }
    …
}
```

```
public class Student extends Person{
    private int studentName;
    public Student(){
        super();
        …
    }
    …
}
```

Exercise

- What is the meaning of Polymorphism? Please give an example of it.

Answer

- Polymorphism means that a variable of a supertype can refer to a subtype object.
- Example:

```java
public class PolymorphismExample {
  public static void main(String[] args) {
    m(new Student());
  }

  public static void m(Object x) {
    System.out.println(x.toString());
  }
}
```

```java
class Student extends Person {
  public String toString() {
    return "Student";
  }
}

class Person extends Object {
  public String toString() {
    return "Person";
  }
}
```

Supertype variable x can refer to subtype student object

Exercise

- What is the output of the following code?

```
public class ClassExercise {

    public static void main(String[] args){

        System.out.println(Integer.parseInt("13"));
        System.out.println(Integer.parseInt("23",10));
        System.out.println(Integer.parseInt("33",16));

    }

}
```

Answer

13

23

51

Exercise

- Please answer “true” or “false” to the following questions? If you answer “false”, please also explain why.

Q: A class cannot contain both static and non-static methods.

Answer

- False.
- A Class can contain both static and non-static methods. The static method can be invoked by using the Class name dot method name. For example:

```
public class CSYE6200 {
   ...
   public static double getAverage(double first, double second){
          return 0;
   }
}

CSYE6200.getAverage(firstData,secondData);  //Invoke getAverage()
```

Exercise

```java
public class Exercise1 {
    public static void main(String[] args) {
        B b = new B();
        b.m(5);
        System.out.println("i is " + b.i);
    }
}

class A {
    int i;

    public void m(int i) {
        this.i = i;
    }
}

class B extends A {
    public void m(String s) {
    }
}
```

A. The program has a compilation error, because m is overridden with a different signature in B.

B. The program has a compilation error, because b.m(5) cannot be invoked since the method m(int) is hidden in B.

C. The program has a runtime error on b.i, because i is not accessible from b.

D. The method m is not overridden in B. B inherits the method m from A and defines an overloaded method m in B.

Answer

D

Exercise

- Write a method that is passed a `String` array and returns the longest string (the one with the most characters). If there are multiple `String` objects that are tied for the longest, return the last such `String`. Write a main() method to test your method.

Answer

```java
public class ClassExamples {

    public static String findLongest(String[] a) {
        String longest = "";
        int maxLength = -1;
        for (int i = 0; i < a.length; i++) {
            if (a[i].length() >= maxLength) {
                maxLength = a[i].length();
                longest = a[i];
            }
        }
        return longest;
    }

    public static void main(String[] args) {
        String[] testVals = { "abcdef", "stuff", "more stuff", "abcdefghij" };

        String answer = findLongest(testVals);

        System.out.println("The longest string was: " + answer);
    }
}
```

Exercise

- Write a class that represents a book. Every book should have a name, number of pages, and year published. Include an appropriate constructor and an `toString()` method. Write a `main()` method in a separate class to test your book class.

Answer

Book.java:

```java
public class Book {
	private String title;
	private int pageCount;
	private int yearPublished;

	public Book(String bookTitle, int bookPageCount, int bookYearPublished) {
		title = bookTitle;
		pageCount = bookPageCount;
		yearPublished = bookYearPublished;
	}

	public String toString() {
		String output = "Book: '" + title + "'";
		output += String.format(", Published: %d", yearPublished);
		output += String.format(", Length: %d pages", pageCount);
		return output;
	}
}
```

ClassExamples.java:

```java
public class ClassExamples {
	public static void main(String[] args) {
		Book hobbit = new Book("The Hobbit", 310, 1937);
		Book iRobot = new Book("I, Robot", 272, 1950);

		System.out.println(hobbit);
		System.out.println(iRobot);
	}
}
```

Exercise

- Write a class that represents a planet. Every planet has a radius and a distance from the sun. Include an appropriate constructor, a method that lets you set the distance separately, a method that returns the approximate volume of the planet (volume of a sphere is $4\pi r^3/3$), and a `toString()` method. Write a `main()` method in a separate class to test the planet class.

Answer

Planet.java:

```
public class Planet {
	private double radius;
	private double distanceFromSun;
	public Planet(double planetRadius, double planetDistance) {
		radius = planetRadius;
		distanceFromSun = planetDistance;
	}
	public void setDistance(double planetDistance) {
		distanceFromSun = planetDistance;
	}
	public double volume() {
		return (4.0/3.0) * Math.PI * Math.pow(radius, 3);
	}
	public String toString() {
		String output = String.format("Radius: %.0f km, ", radius);
		output += String.format("Distance From Sun: %.0f km, ", distanceFromSun);
		output += String.format("Approximate Volume: %.0f cubic km", volume());
		return output;
	}
}
```

ClassExamples.java:

```
public class ClassExamples {
	public static void main(String[] args) {
		Planet earth = new Planet(6371, 150000000);
		System.out.println(earth);
		earth.setDistance(149597871);
		System.out.println(earth);
	}
}
```

Exercise

- Create a class Android whose objects have unique data. The class has the following attributes:
  - tag: a static integer, begin at 1, change each time an instance is created.
  - name: a string that is unique for each instance of this class
- Android class has the following methods:
  - Android: default constructor that sets name to “Bob”+tag
  - getName: return the name portion of the invoking object
  - isPrime(n): a private static method, return true if n is prime
  - changeTag: a private static method, replace tag with next prime number

Answer

```
public class Android {

    private static int tag = 1;
    private String name;

    public Android() {
        name = "Bob"+tag;
        changeTag();
    }

    private static void changeTag(){
        int tryNext = tag + 1;
        while(!isPrime(tryNext))
            tryNext = tryNext + 1;
        tag = tryNext;
    }

    private static boolean isPrime(int n){
        boolean hasFactor = false;
        for(int factor = 2; factor<n; factor++){
            if(n%factor == 0)
                hasFactor = true;
        }
        return !hasFactor;
    }

    public String getName(){
        return name;
    }

}
```

Exercise

- Continued. Create a program that tests the class Android described above.

Answer

```
public class Android {

  …
  public static void main(String[] args) {
    for(int count = 1; count<10; count++){
        Android myAndroid = new Android();
        System.out.println("Created an android with name " +
                                  myAndroid.getName());
    }
  }
}
```

Exercise

- The following method compiles and executes but does not work as you might hope. What is wrong with it? How do you correct it?

```
public static int[] copyArray(int[] anArray){

    int[] temp = new int[anArray.length];

    temp = anArray;

    return temp;

}
```

Answer

- The method does not return an array distinct from the given argument array. Rather, it returns a reference to the array it is given.
- To make a duplicated array, you would replace the statement “temp = anArray” with

```
for(int i = 0; i<anArray.length; i++){
    temp[i] = anArray[i];
}
```

Exercise

- Please use ArrayList to contain four greetings and print out all greetings at the end, such as:

```
All greetings

Hello! How are you?
Hi man, what's up!
Yo! How's going?
Hey! How are you doing?
```

Answer

```
public class ClassExample {

    public static void main(String[] args){

        ArrayList<String> greetings = new ArrayList<String>();

        greetings.add("Hello! How are you?");
        greetings.add("Hi man, what's up!");
        greetings.add("Yo! How's going?");
        greetings.add("Hey! How are you doing?");

        System.out.println("All greetings");
        System.out.println("");
        for(String s: greetings){
            System.out.println(s);
        }
    }
}
```