INFO1103: Introduction to Programming
Semester 1, 2015 - Study Quiz

Eric Liu

June 23, 2015

Week 1

Running Java Programs

1. What is the difference between the `java` and `javac` commands?
2. Where inside a Java program does execution begin?
3. If your Java class looks like the following:

```
public class Pikachu {
    public static void main(String[] args){
        System.out.print("Raichu");
        System.out.println("Zap");
    }
}
```

   What must this file be named, including the file type extension?

4. If the above code is executed in a command-line terminal, what is shown?

Types of Errors

1. What is the difference between a run-time and compile-time error?
2. Consider the following code:

```
public class Pokemon {
    public static void main(String[] args){
        System.out.println("Go! Charmander!")
    }
}
```

   What kind of error occurs in the above code?

3. How can the above code be changed to run?

Week 2

Binary and Hexadecimal Numbers

1. How would the number 11 be represented in binary?
2. What are the two values a bit can be?
3. How would the number 27 be represented in hexadecimal?
4. How many bits are there in a byte?
5. Why do computers work well with binary instead of decimal?

Variables

1. What is the purpose of a variable?
2. List the eight primitive types in Java.
3. Write the line of code which stores the value 25 inside an integer variable called `pikachu`.
4. Write the line of code which updates the above `pikachu` variable to store the value of 100.
5. Write the line of code which stores the phrase "Rawr!" in a String variable called `charmander`.

The Scanner class

1. Write the line of code which creates a Scanner object called `reading` that reads keyboard input, excluding import statement.
2. Write the line of code which reads in a single word using the previous Scanner and stores it in a String variable called `thisWord`.
3. Write the line of code which reads in a double-precision number using the previous Scanner and stores it in a double variable called `thisValue`.

Command-line Arguments

1. Consider the following code:

```
public class Woof {
    public static void main(String[] args){
        System.out.println("==" + args[0] + " " + args[1] + "==");
    }
}
```

   What is printed when the above code is compiled then executed with `java Woof super meowzers`?

Arithmetic Operations

1. Consider the following code:

```
public class Hoenn {
    public static void main(String[] args){
        int cities = 10;
        int upgraded = 2 * (++cities);

        int towns = 10;
        int upgrading = 2 * (towns++);
        System.out.println(upgraded + " " + upgrading);
    }
}
```

   What is printed by the program?

2. What is the value of `towns` at the end of the program?
3. What is the result of `15%4 + 10%10` ?
4. What is the result of `10-->2` ?
5. Write the line of code that calculates 7^5 and stores it in an integer variable called `magikarp`.

Week 3

Boolean Logic

1. List the two other logical operators besides `&&`.
2. List the four other comparison operators besides `==` and `<=`.
3. Consider the following code:

```
public class Inequality {
    public static void main(String[] args){
        int x = 10;
        boolean result;
        // Store in the result variable, the output of 0 < x < 10
        System.out.println(result);
    }
}
```

   Write the line of code which performs the above comment's function.

4. For the above program, what is printed?
5. What is the difference between `=` and `==`?

Casting

1. Describe the purpose of casting.
2. What is the result of `5/2.0`?
3. What is the result of `5/2`?
4. What kind of rounding occurs in Java when floating-point values are casted to integers.

`if-else` Statements

1. Describe the syntax of a basic `if-else` statement.
2. Consider the following code:

```
public class ToHealOrNotToHeal {
    public static void main(String[] args){
        int maxHP = 150;
        int currHP = 100;

        if (currHP == maxHP) {
            System.out.println("Full HP!");
        } else {
                if (currHP < maxHP/2) {
                        System.out.println("Hurt, you should heal!");
                } else {
                        System.out.println("Hang in there!");
                }
        }
    }
}
```

   What is printed by the above program?

3. Can the `else` component be omitted when writing `if-else` statements?
4. What does nesting `if`-statements allow us to do?
5. What do `else if` statements allow us to do?

Deskchecks

1. Consider the following code again:

```
public class ToHealOrNotToHeal {
    public static void main(String[] args){
        int maxHP = 150; // Keep fixed
        int currHP = 100; // Test when equal to: 100, 150, 75, 74, 151

        if (currHP == maxHP) {
            System.out.println("Full HP!");
        } else {
                if (currHP < maxHP/2) {
                        System.out.println("Better heal!");
                } else {
                        System.out.println("Hang in there!");
                }
        }
    }
}
```

   Perform a deskcheck to assess the output of the above code with `currHP` at the varying values.

`while` Loops

1. Describe the syntax of a `while` loop.
2. List in three or four steps, the actions of a `while` loop.
3. Consider the following code:

```
public class StepCount {
    public static void main(String[] args){
        int stepCount = 0;

        while(stepCount < 3){
            System.out.println(stepCount);
        }
    }
}
```

   What will the above program print?

Week 4

`for` Loops

1. Describe the syntax of a `for` loop.
2. Write code that makes a `for` loop print "Meow" on new lines infinitely.
3. Consider the following code:

```java
public class TrainerCard {
    public static void main(String[] args){
        for(int i = 0; i < 5; i++);{
            System.out.println("Pokemon!!");
        }
    }
}
```

Why is the phrase "Pokemon!!" only printed once instead of five times?

`do-while` Loops

1. Describe the syntax of a `do-while` loop.
2. List in three or four steps, the actions of a `do-while` loop.
3. What is the main difference between `while` and `do-while` loops?

Terminating Loops

1. Describe the purpose of the `continue` keyword.
2. Consider the following code:

```java
public class Logic {
    public static void main(String[] args){
        int a = 0;

        // A
        while(a < 1){
            // B
            while(true){
                break;
            }
            // C
            a++;
        }
        // D
        System.out.println("Mew was here!");
    }
}
```

When the `break` statement is executed, which section of code is executed next? (A, B, C or D)

3. Write a program that requests a String input from the user, prints the inputted String and repeats until the word "woof" is entered. Here is an example scenario:

```
Enter a word: quit
You entered ... quit
Enter a word: quit
You entered ... quit
Enter a word: woof
Okay!
```

Arrays

1. Describe the syntax of creating (declaring and initialising) an array.
2. Write the lines of code that create a double array called `fancy` and store two values: 0.21 and 1.31.
3. Consider the following code:

```
public class PokemonParty {
    public static void main(String[] args) {
        String[] pokemonList;
        pokemonList[0] = "Bulbasaur";
    }
}
```

Is there an error in the above program? If so, what is it?

Week 5

Methods

1. Describe the syntax of a method excluding modifiers.
2. Consider the following code:

```
public class CatchPokemon {
    public static void main(String[] args) {
        if(throwPokeBall()){
            // A
            System.out.println("Gotcha! The Pokemon was caught!");
        } else {
            // B
            System.out.println("Oh no! The Pokemon broke free!");
        }
        // C
    }

    public static boolean throwPokeBall(){
        // D
        if(Math.random() > 0.5){
            return true;
        } else {
            return false;
        }
    }
}
```

Which section of code is executed after `throwPokeBall()` is called from the `main` method? (A, B, C or D)

3. For the above program, if `throwPokeBall()` returns false, what is printed?
4. What does the `void` return type signify?
5. List two benefits of using methods?

Passing Parameters

1. Write a method with the following signature `public static boolean isOdd(int n)`, it should return `true` if the passed parameter is an odd number, `false` otherwise.

2. Write a program that requests the user to type an integer and print the square. Use the following method signature in your program: `public static int square(int n)`. The program should loop until a negative value is entered.
Below is an example scenario:

```
Enter an integer: 5
Square is ... 25
Enter an integer: 10
Square is ... 100
Enter an integer: -2
Okay!
```

3. Consider the following code:

```
public class PokemonBattle {
    public static void main(String[] args) {
        boolean isBattle = true;
        String currentPokemon = "Gardevoir";

        if(isBattle){
            activateMega(currentPokemon);
        }
        System.out.println(currentPokemon);
    }

    public static void activateMega(String pokemon){
        pokemon = "Mega " + pokemon;
    }
}
```

Explain why the output of the above program is "Gardevoir" and not "Mega Gardevoir".

Variable Scope

1. Consider the following code:

```
public class SilphCo {
    static int x = 10;
    public static void main(String[] args) {
        int y = 2;
        for(int i = 5; i < 10; i++){
            // A
        }
        // B
    }

    public static void useSilphScope(){
        for(int j = 0; j < 100; j++){
            // C
        }
        // D
    }
}
```

List the sections where variable `x` can be seen. (A, B, C and/or D)

2. Similarly, for the above program, list where `y` can be seen.
3. Similarly, list where `i` can be seen.
4. Similarly, list where `j` can be seen.

Strings

1. Write the line of code which declares and initialises an empty String variable called `blank`.
2. Consider the following code:

```
public class PokemonBattle {
    public static void main(String[] args){
        String wildPokemon = "Zubat";
        System.out.println("A wild " + wildPokemon.toUpperCase() + " appears!");
    }
}
```

What is printed by the program?

3. For the above program, what is the returned value of `wildPokemon.substring(1,3)`?
4. Write the line of code that prints the phrase "true\false".
5. Why is `.equals()` used to compare equality of Strings as opposed to `==`?

The StringBuilder class

1. What is the difference between a String and a StringBuilder object?
2. When would you typically use a StringBuilder instead of a String?
3. Consider the following code:

```
public class PokemonBattle {
    public static void main(String[] args){
        StringBuilder woof = new StringBuilder("Woof");
        woof.append(" to infinity and beyond!");
        // Print contents of StringBuilder object here
    }
}
```

Write the line of code that performs the above comment's function.

Week 6

Exceptions and Exception Handling

1. Give the definition of an `Exception`?
2. What is the difference between a checked and unchecked exception?
3. Write code that will generate a `NullPointerException`.
4. Consider the following code:

```
public class Glitch {
    public static void main(String[] args){
        int pokedexNo = 0;
        if(pokedexNo > 0){
            System.out.println("Wild Pokemon!");
        } else {
            throw new Exception("Missingno!");
        }
    }
}
```

Write the above code again after modification in the following two scenarios:

(a) Handle the `Exception` with a `try-catch` clause

(b) Handle the `Exception` with a `throws` keyword

5. List three common predefined `Exception` objects you have come across. (E.g. `NullPointerException`)
6. (*Requires knowledge of Inheritance*) Explain why more specific `Exception` objects must be placed before more general ones.

Reading/Writing Files

1. When opening a file, which exception has a chance to be thrown?
2. Write a program that prints the contents of a text file called "Jolteon.txt" line-by-line. The contents of the "Jolteon.txt" are listed below:

```
Dear Trainer,
I would like to try Slowpoke Tail sometime.
When are we going to the Safari Zone again?
The bait was really tasty!
P.S. Don't eat yellow snow!
-- Jolteon
```

3. When you have finished writing to a text file, what must you not forget to do?
4. Write a program that requests the user to enter a sentence, then output a text file called "awesome.txt" which contains the input sentence.
Below is an example scenario:

```
Enter a sentence:
Rayquaza wears lipstick!
```

A text file called "awesome.txt" should be created and contain the above sentence.

Week 7

Classes and Objects

1. List two benefits of using objects over primitives.
2. How do objects and classes relate to each other?
3. Describe the syntax of a class.
4. Write the code for a class called `Item` and declare three *instance variables*, a String `name`, a String `description` and an integer `quantity`.

Constructors

1. Describe the purpose of a constructor.
2. Describe the syntax of a constructor.
3. Consider the following code:

```
public class Pikachu {
    String name;
    int HP, ATK, SPA, DEF, SPD, SPE;

    // Add a constructor here
}
```

   Write the constructor that initializes the default Pikachu object to the following values:

   (a) Name is "PIKACHU"
   (b) HP is 10
   (c) ATK is 8
   (d) SPA is 14
   (e) DEF is 6
   (f) SPD is 7
   (g) SPE is 20

4. Consider the following code:

```
public class Player {
    String name;

    public Player(String name) {
        name = name;
    }
}
```

   Describe how the above program can be fixed.

Static vs. Non-static

1. What does the static keyword denote?
2. Consider the following phrase: "Every `Person` object has a `name`."
   Should `name` be a static or non-static variable and why?
3. Consider the following phrase: "Every `Pikachu` object has an `evolvesTo` variable which is equal to 'Raichu'."
   Should `evolvesTo` be a static or non-static variable and why?
4. What is the difference between a "class variable" and an "instance variable"?
5. What is the rule regarding the types of variables *static methods* and *non-static methods* can refer to?

Enumerations

1. Give the definition of an enumeration.
2. List a benefit of using enumerations.
3. Consider the following code:

```
class Lost {
    /* North = 1, South = 2, East = 3, West = 4 */
    public void travel(int direction) {
        switch(direction) {
            case 1:
                System.out.println("Keep going!");
                break;
            case 2:
                System.out.println("That's backwards!");
                break;
            case 3:
            case 4:
                System.out.println("Wrong way!");
                break;
        }
    }
}
```

Create an enumeration of directions to prevent incorrect values from being entered and implement them in the Lost class.

Encapsulation

1. Give the definition of encapsulation.
2. What is the difference between the `public` and `private` access modifiers?
3. Consider the following code:

```
public class MayProfile {
    public String name = "May";

    public String getName(){
        return this.name;
    }
}
```

Why is the above code problematic in regards to encapsulation?

Week 8

More Constructors and Methods

1. Consider the following code:

```
public class TrainerCard {
    private int id;

    public TrainerCard(){
        id = 111;
```

```
    }

    public TrainerCard(int id) {
        this.id = id;
    }
}
```

What is a benefit of having multiple constructors?

2. Consider the following code:

```
public class PokemonMath {
    public int sum(int a, int b){
        // returns result as an integer
    }

    public int sum(int a, int b, int c){
        // returns result as an integer
    }

    public String sum(int a, int b){
        // returns result as a String
    }
}
```

What is the problem with the above code?

Testing

1. Give definitions for the following terms:
   (a) Testing
   (b) Test-driven Development
   (c) Execution path
   (d) Black-box Testing
   (e) White-box Testing
   (f) Regression Testing
   (g) Assertion
   (h) Precondition
   (i) Postcondition
   (j) Class invariant
2. Consider the following code:

```
class Magikarp {
    private int level;

    public Magikarp(int level) {
        this.level = level;
    }

    public void useRareCandy() {
        ++level;
    }
```

```
    public boolean canEvolve() {
        if (level >= 20)
            return true;
        return false;
    }
}
```

Write a tester class called `MagikarpTester` that instantiates a level 19 Magikarp, uses a rare candy on it once and then tests whether it can evolve. If it can evolve, print "Passed!", if it cannot evolve, print "Failed!".

Week 9

Designing Classes

Recommended: Redo Week 10 tutorial questions.

Week 10

Inheritance

1. Describe a benefit of using inheritance.
2. Give a real-life example of inheritance.
3. If class Meow extends Woof, which class is a subclass and which is a superclass.
4. What is the effect of the `protected` modifier?
5. Consider the following code:

```
class Bulbasaur {
    public int level;
    public String name, type;
    public int[] stats;

    public void useTackle(){
        System.out.println("It did 8 damage!");
    }
}

class Charmander {
    public int level;
    public String name, type;
    public int[] stats;

    public void useScratch(){
        System.out.println("It did 10 damage!");
    }
}
```

Describe a solution, using inheritance, to minimise the repeated code within the `Bulbasaur` and `Charmander` classes.

6. Inside a subclass's constructor, how can you call the superclass's constructor?

Polymorphism

1. Give a definition of polymorphism.
2. What is a benefit of using polymorphism?
3. Is there a syntax for polymorphism, why?
4. What is the effect of the `@Override` keyword?
5. What is the difference between *method overriding* and *method overloading*?
6. Consider the following code:

```
class Animal {
    public void talk(){
        System.out.println("Graah");
    }
}

class Dog extends Animal {
    @Override
    public void talk(){
        System.out.println("Woof!");
    }
}

class Cat extends Animal {
    @Override
    public void talk(){
        System.out.println("Meow!");
    }
}

public class PolymorphismTester {
    public static void main(String[] args){
        // Write code here that demonstrates polymorphism with the above classes
    }
}
```

Write code that performs the above comment's function.

Common Array Algorithms

1. Consider the following code:

```
public class PokedexEntry {
    public static void main(String[] args) {
        String[] latios = new String[4];
        latios[0] = "Latios";
        latios[1] = "Type: Dragon/Psychic";
        latios[2] = "Species: Eon";
        latios[3] = "Description: Such fast! Much wow!";
        printReverseEntry(latios);
    }

    public static void printReverseEntry(String[] entry){
        // Print each element of array on a new line in reverse order
    }
}
```

Write the code that performs the function as specified in the comment above.

2. Consider the following code:

```
public class Search {
    public static void main(String[] args) {
        int[] itemQuantities = {11, 21, 8, 10};

        System.out.println("Smallest quantity is: " + findMin(itemQuantities));
    }

    public static int findMin(int[] array){
        // Find and return the smallest value in the array
    }
}
```

Write the code that performs the above comment's function.

3. Consider the following code:

```
public class PokemonBox {
    private String[] currentBox = {"Charmander",
                                   "Bulbasaur",
                                   "Pikachu",
                                   "Squirtle" };

    public static void main(String[] args){
        PokemonBox box = new PokemonBox();
        System.out.println(box.toString());
    }

    @Override
    public String toString(){
        // Concatenates all elements inside the currentBox array, separated by a space
    }
}
```

Write the code the performs the above comment's function. Below is the expected output of the program:

```
Charmander Bulbasaur Pikachu Squirtle
```

Week 11

Primitive Wrapper Classes

1. List two benefits of a wrapper class over a primitive.
2. List two methods/variables of the Integer wrapper class.

The `ArrayList`

1. Describe the syntax for creating an `ArrayList`.
2. What a benefit of using an `ArrayList` over an array?
3. List the statements which do the following on an `ArrayList`:

(a) get the element at index i

(b) remove the element at start of list

(c) get the current size of list

(d) add an element `meow` to end of list

4. Consider the following code:

```
public class AccountTester {
    public static void main(String[] args){
        ArrayList<Account> list = new ArrayList<Account>();
        Account[] array = new Account[10];

        Account kitty = new Account("Meow", "burgers");
        Account puppy = new Account("Woof", "cupcakes");

        // Add both accounts to the ArrayList and array
        // Then, print kitty's name and secret from the ArrayList
        // Also, print puppy's name and secret from the array
    }
}

class Account {
    public String name;
    private String secret;

    public Account(String name, String secret){
        this.name = name;
        this.secret = secret;
    }

    public String getSecret(){
        return this.secret;
    }
}
```

Write the code that performs the above comment's function. Below is the expected output:

```
Meow likes burgers.
Woof likes cupcakes.
```

Sets and Maps

1. What is the primary difference between Sets and Lists?
2. What is the main difference between a TreeSet and a HashSet?
3. What is the primary feature of a Map?
4. What is the constraint of keys in a Map?

`for-each` Loop

1. Describe the syntax of the `for-each` loop.
2. What is the benefit of using a `for-each` loop?
3. What is a limitation of the `for-each` loop?

4. Consider the following code:

```
public class PokeCentre {
    public static void main(String[] args){
        ArrayList<Pokemon> party = new ArrayList<Pokemon>(6);
        // Imagine 6 Pokemon objects are added to the party list

        // Convert the below loop
        for(int i = 0; i < party.size(); i++){
            Pokemon current = party.get(i);
            current.heal();
        }
    }
}

class Pokemon {
    public int maxHP = 100;
    public int HP = 20;

    public void heal(){
        HP = maxHP;
    }
}
```

Replace the `for` loop with a `for-each` loop that performs the same function.

Recursion

1. What is an advantage of using recursion?
2. What is a disadvantage of using recursion?
3. List the two essential ingredients for recursion.
4. Write a method that recursively computes a factorial and uses the following method signature: `public static int factorial(int n)`. (Obligatory example)
5. Write a method that *iteratively* computes a factorial, using the same method signature as above.
6. Consider the following information:

```
Fibonacci sequence: 0, 1, 1, 2, 3, 5, 8, 13, 21, ...

fibonacci(0) -> 0
fibonacci(1) -> 1
fibonacci(2) -> 1
```

Fibonacci numbers defined by the sum of the two previous values (starting from 0 and 1, base cases). Assuming n is always positive, write a method which determines the nth Fibonacci number recursively. Use the following method signature: `public static int fibonacci(int n)`

Week 12

Abstract Classes

1. What the purpose of abstract classes?
2. Give a real-life example of an abstract class and some of its subclasses.

3. Consider the following code:

```
class PlayerCharacter {
    private int x, y;

    public void move(int newX, int newY){
        this.x = newX;
        this.y = newY;
    }

    public int getX(){
        return this.x;
    }

    public int getX(){
        return this.y;
    }
}

class NonPlayerCharacter {
    private int x, y;
    private int pushback = 5;

    public void move(int newX, int newY){
         this.x = newX - pushback;
         this.y = newY - pushback;
    }

    public int getX(){
        return this.x;
    }

    public int getX(){
        return this.y;
    }
}
```

Write an abstract class called `Character` such that extending this abstract class removes redundant code and groups together the above two classes.

Java Interfaces

1. What is the purpose of a Java interface?
2. What is the required by a class that implements an interface?
3. Describe a benefit of using interfaces in regards to arrays or lists.
4. Write an interface called `Talkable` that contains a method with the signature `void talk()`.
5. Describe the differences between abstract classes and interfaces for the following properties:
   (a) Instantiable?
   (b) Methods?
   (c) Variables?
   (d) Constructors?

2D Arrays

1. Describe a purpose for using 2D arrays.
2. Is there a limit to the number of dimensions an array can have?
3. Consider the following code:

```
public class PokemonStatistics {
    public static void main(String[] args) {
        int[][] rawStats = {{5, 4, 5},
                            {1, 2},
                            {10, 11, 4}};

        int[] summedStats = summate(rawStats);

        for(int i = 0; i < summedStats.length; i++){
            System.out.println(summedStats[i]);
        }
    }

    public static int[] summate(int[][] data){
        // Calculate the sum of each row of data and store it in a new array
    }
}
```

Write the code the performs the above comment's function. Below is the expected output of the program:

```
14
3
25
```

`switch` Statements

1. Describe the syntax of a `switch` statement with two cases and a default case.
2. Consider the following code:

```
public class FireTypeMatchup {
    public static void main(String[] args){
        String defendingType = "Water";

        switch(defendingType){
            case "Grass":
            case "Ice":
            case "Bug":
                System.out.println("It's super effective!");
                break;
            case "Fire":
            case "Water":
            case "Ground":
                System.out.println("It's not very effective.");
                break;
            default:
                System.out.println("It hit!");
        }
    }
```

```
}
```

What is printed by the program?

3. For the above program, what is printed if `defendingType` is "Normal"?

Credits and Thanks to:

- Irena Koprinska, Masa Takatsuka, John Stavrakakis (Lecturers) - For making this awesome programming course!
- Darren Shen, Scott Sidwell, Ryan Dicembre (Tutors) - For tons of advice and recommendations for questions!
- SUITS - For providing the resources to help run the workshops!
- Gabe Setiawan, Chris Chen - For all the help!
- Pokemon - Because Pokemon! :D

Some people want it to happen,
Some wish it would happen,
Others make it happen.

Michael Jordan