Summary

1 Disclaimer

2 Methods and Data
   2.1 Static vs. Non-Static
   2.2 Static
   2.3 Non-Static
   2.4 Calling Static v Non-Static Methods
   2.5 Pass-by-Value vs Pass-by-Reference

3 Arrays
   3.1 1D Arrays
      3.1.1 General Syntax
      3.1.2 Printing Arrays
   3.2 2D Arrays
      3.2.1 General Syntax
      3.2.2 Printing
   3.3 Reminders and Warnings

4 Objects
   4.1 Instantiating Objects
   4.2 Constructors

5 Practice Written Exam
   5.1 Short Answer
   5.2 Tracing

6 General Suggestions for the Exam
   6.1 Written Exam
   6.2 Programming Exam
1 Disclaimer

This is a review of this courses’ material, but there may be material on the exam not covered in this study guide.

2 Methods and Data

2.1 Static vs. Non-Static

Static methods belong to the class and only have one copy of the information. For example, a Clock class should be static, because if you change something on a clock you want it change in all other objects too. Static methods are used when you need to use them when you aren’t going to use instance variables.

Non-Static methods are instances of the class, so you can manipulate instance variables. For example, a Student class should be non-static because you want to have all of your objects be different (different name, id, major, etc)

Note on calling methods: The only time you need to create an object of the class (for example, Review rv = new Review ( ) ) is when you call a non-static method in a static method (a common call is in the main method, which is static). If you were to have two non-static methods or a non-static method calling a static method you wouldn’t need to create an object.

2.2 Static

    // Example of a static class: Clock

    public class Clock {  
        private static int hour, minute;

        public Clock ( int h, int m){
            hour = h;
            minute = m;
        }
        // default time if no hour and minute are given
        public Clock (){  
            hour = 12;
            minute = 0;
        }
        public static void increaseHour ( int num ){
            hour += num ;
        }
        public static void increaseMinute ( int num ){
            minute += num ;
        }
        // Calling above methods to save work
        public static void increaseHour (){
            increaseHour (1);
        }
        public static void increaseMinute (){  
            increaseMinute (1);
        }
        public String toString (){  
            return String . format ("%02d:%02d", hour, minute ) ; // returns time with 00:00 format.
        }
    }
import java.util.Arrays;

/* Example of a Non-Static Class: Student */

public class Student {
    // instance variables
    private String name, year, major, minor;
    private int id;

    // constructor
    public Student (String _name, String _major, String _year, int _id){
        this.name = _name;
        major = _major;
    }
}
public Student (String _name, String _major, String _minor, String _year, int _id) {
    name = _name;
    major = _major;
    minor = _minor;
    year = _year;
    id = _id;
}

public void increaseYear (){
    switch (year) {
        case "Freshman": year = "Sophomore"; break;
        case "Sophomore": year = "Junior"; break;
        case "Junior": year = "Senior"; break;
        case "Senior": year = "Super Senior"; break;
        default: year = "Unknown"; break;
    }
}

public void changeMajor (String new_major) {
    major = new_major;
}

public void addMinor (String _minor) {
    minor = _minor;
}

public String toString () {
    return String.format("Name: %s\nMajor: %s\nMinor: %s\nYear: %s\nID: %d", name, major, minor, year, id);
}

public static void main (String [] args) {
    Student bob = new Student("Bobby Joe", "Mathematics", "Computer Science", "Senior", 90314);
    Student john = new Student("John Doe", "Computer Science", "Freshman", 90213);
    Student julie = new Student("Julie Sparkles", "English", "Junior", 91942);
    Student steve = new Student("Steve Reeves", "Physics", "Computer Science", "Junior", 90870);
    //System.out.println(bob);
    Student [] cs160 = {bob, john, julie, steve};
    System.out.println("Total Students in CS160:
    for (int i = 0; i < cs160.length; i++) {
        System.out.println(cs160[i] + "\n");
    }
    julie.changeMajor("Computer Science"); // because it’s awesome
    john.addMinor("English");
    steve.increaseYear();
    System.out.println("Updated Total Students:\n");
    for (int i = 0; i < cs160.length; i++) {

System.out.println(cs160[i] + "\n");
}
}

/* Console Output:

Total Students in CS160:

Name: Bobby Joe
Major: Mathematics
Minor: Computer Science
Year: Senior
ID Number: 90314

Name: John Doe
Major: Computer Science
Minor: None
Year: Freshman
ID Number: 90213

Name: Julie Sparkles
Major: English
Minor: None
Year: Junior
ID Number: 91942

Name: Steve Reeves
Major: Physics
Minor: Mathematics
Year: Sophomore
ID Number: 90870

Updated Total Students:

Name: Bobby Joe
Major: Mathematics
Minor: Computer Science
Year: Senior
ID Number: 90314

Name: John Doe
Major: Computer Science
Minor: English
Year: Freshman
ID Number: 90213

Name: Julie Sparkles
Major: Computer Science
Minor: None
Year: Junior
ID Number: 91942

Name: Steve Reeves
Major: Physics
Minor: Mathematics

*/
2.4 Calling Static v Non-Static Methods

Static Methods:

```java
public static void main (String [] args) {
    int [] myArray = {1, 2, 3, 4};
    printMyArray(myArray);
}

public static void printMyArray(int [] array){
    for (int i = 0; i < array.length; i++){
        System.out.print(array[i] + " ");
    }
    System.out.println(); //used for spacing
}
```

Non-Static Methods:

```java
public class Review {
    private int [] myArray = {1, 2, 3, 4};
    public static void main (String [] args) {
        Review rv = new Review ();
        rv.printMyArray ();
    }

    public void printMyArray (){
        for (int i = 0; i < myArray.length; i++){
            System.out.print(myArray[i] + " ");
        }
        System.out.println(); //used for spacing
    }
}
```

2.5 Pass-by-Value vs Pass-by-Reference

Pass-by-Reference are usually objects. This is because they have their own specified memory (aka it has memory allocated for the variable), so when a method calls that variable it accesses that place in memory and manipulates that. Therefore, Pass-by-Reference variables ARE CHANGED! For example:

```java
public static void main (String [] args0){
    int [] intArray = {1, 2, 3};
    System.out.println(Arrays.toString(multiplyIndex0(intArray,9)));
    //prints [9,2,3]
    System.out.println(Arrays.toString(intArray));
    //prints [9,2,3]
}

public static int [] multiplyIndex0(int [] i, int p){
    i[0] = p;
    return i;
}
```

Pass-by-Value are primitive types that are passed into a method’s parameter. THESE VALUES ARE NOT CHANGED OUTSIDE THE METHOD THAT INITIALIZES THEM. The calling method creates a copy of the values so the original values are never changed. For example:
//method that is calling (aka caller method)
public static void main (String [] args0) {
    int number = 100;
    increment(number);
    System.out.println("Number: " + number);
}

//method being called (aka calling method)
public static void increment(int n){
    n++;
}

//NUMBER IS NEVER CHANGED

3  Arrays
3.1  1D Arrays
3.1.1  General Syntax

    //two ways to initialize an array
    //you know only the size
    typeOfArray [] nameOfArray = new typeOfArray [sizeOfArray];
    //you know what the values are
    typeOfArray [] nameOfArray1 = {values, you, want, in, the, array};

• To manipulate the array

    String [] sArray = new String [3];
    //to change on value in the array
    sArray[0] = "Hola";
    //to find the length of a string
    int [] iArray = {1, 1, 1, 1, 1, 0};
    //remember there are no parenthesis after length like there is with
    //Strings.
    System.out.println(iArray.length);

Initializing a 1-D Array:

    int [] iArray = new int [3];
    String [] csClasses = {"CS160", "CS161", "CS200", "CS270", "CS253"};

Manipulating 1-D Arrays:

    int [] iArray = new int [10];

    //assigning all indexes to one value
    for (int i = 0; i < iArray.length; i++)
        iArray[i] = 1;

    //changing a value at a specific index
    iArray[3] = 5;

    //getting length
    //could also print length using arrayName.length;
    int arrayLength = arrayName.length;
3.1.2 Printing Arrays

```java
for (int i = 0; i < arrayName.length; i++) {
    // Warning: Read directions!
    // You be asked to print on different lines, on the same lines,
    // with spaces in between, with a comma between, etc.
    System.out.print(arrayName[i]);
}
// another way to print:
// Warning: On programming exam you may need to import.
// So either type Control + Shift + O or type
// import java.util.Arrays; (do this outside of your class)
System.out.println(Arrays.toString(arrayName));
```

3.2 2D Arrays

3.2.1 General Syntax

```java
int[][] iArray = new int[3][3];
String[][] sArray = {{"Hi", "there"}, {"How", "are", "you?"}};
```

Manipulating 2-D Arrays:

```java
int[][] board = new int[3][3];

// assigning all indexes to one value
for (int row = 0; row < board.length; row++)
    for (int col = 0; col < board[row].length; col++)
        board[row][col] = 0;

// changing one value at a specific index
board[0][2] = 1;
```

3.2.2 Printing

```java
for (int row = 0; row < arrayName.length; row++) {
    for (int col = 0; col < arrayName[row].length; col++) {
        // Again read directions
        System.out.println(arrayName[row][col]);
    }
}
```

3.3 Reminders and Warnings

Be careful with your indexes. If a 2-D Array has a length of 3 and a height of 3, remember when you print or change the values that you could only use indexes 0 - 2.

Some common exceptions:

ArrayIndexOutOfBoundsException: To fix check all of your loop ranges and all the places that you changed a value (i.e. `array[3] = 3;`). Make sure you are never trying to access any index greater than or equal to the array length (same concept for 2-D Arrays).

NullPointerException: To fix check to make sure your array has been initialized.
public class Book {
    // Instance Variables
    private String title;
    private String author;
    private int year;

    // Constructor
    // NOTE: public Book (method name must be the exact same
    // as class name. You are not returning anything so the
    // format is just public name (parameters, if, needed){}
    public Book (String _title, String _author, int _year) {
        title = _title;
        author = _author;
        year = _year; // NOTE: no return value
    }

    // Getters
    public String getTitle (){
        return title;
    }
    public String getAuthor () {
        return author;
    }
    public int getYear () {
        return year;
    }

    // Setters
    public void setTitle (String _title){
        title = _title;
    }
    public void setAuthor (String _author) {
        author = _author;
    }
    public void setYear (int _year) {
        year = _year;
    }

    // toString
    public String toString () {
        String s = "";
        s += "Title: " + title + ", ";
        s += " Author: " + author + ", ";
        s += " Year: " + year;
        return s;
    }

    public static void main (String [] args) {
        Book book0 = new Book ("It\'s Raining from the Clouds", "Oh Knowledgeable One", 1970);
        Book book1 = new Book("Life Without a Cell Phone: The Nightmare of Tweens", "Bored and Social", 2013);
        Book book2 = new Book ("Running out of Clever Names", "Addy Moran", 2016);
        Book [] Library = {book0, book1, book2};
```java
for (int i = 0; i < Library.length; i++)
    System.out.println(Library[i]);
```

Console Output:
Title: It’s Raining from the Clouds, Author: Oh Knowledgeable One, Year: 1970
Title: Life Without a Cell Phone: The Nightmare of Tweens, Author: Bored and Social, Year: 2013
Title: Running out of Clever Names, Author: Addy Moran, Year: 2016

4.1 Instantiating Objects

General Syntax: `ClassName objectName = new ClassName (constructor parameters or can be empty depending on the constructor));`

4.2 Constructors

Purpose of Constructors: In the example, I made a `Book` class that takes a title, author, and year as parameters (using a constructor). By doing this we can attach different kinds of variables together (i.e. `int`, `String`, `double`, etc.) and keep them connected by making an Object based off that input.

General Syntax: `public ClassName (sometimes parameters)`
5 Practice Written Exam

5.1 Short Answer

1. Declare a 4x4 2-D int array, called board.
2. Initialize every value of board to 1.
3. Change the value on the first row, second column to 2.
4. Print board by using for loops.
5. Inside the predefined class Student create a Student object called student0, who’s name is "James Bond", his student id is 007. Use the following code as guidance:

   ```java
   public class Student {
       String id;
       String name = "";
       public Student (String _id, String _name) {
           id = _id;
           name = _name;
       }
   }
   ```

6. Using the same class (Student) and the code from above. Create an Student object called student1, who’s name is ”Jr Bond”, his student id is 008.
7. Create an array of type Student called OverAchievers and insert student0 and student1 (from questions 6 and 7) into the array.
8. Declare a Scanner that reads from the keyboard.
9. Write an if/else if/else statement for the following information (year is already declared):

   - if year is 1970, print "history is cool"
   - if year is 1980, print ”Era of Hippies!”
   - if year is 1990, print ”Cassette Tapes!”
   - if year is 2000, print ”iPhone’s begin their take over...”
   - if year is 2010, print ”US wished Queen Elizabeth II happy birthday on the wrong day, good start to the decade...”
   - if year is anything else, print ”Huh, I’m not sure what to say”

10. Write a switch statement based off the same information from question 12. If given the following code, what is printed?

    ```java
    String s = " Roses are red, violets are blue,..."
    // 11.
    System.out.println(s.substring(0));
    // 12.
    System.out.println(s.substring(6, 8));
    ```

5.2 Tracing

Instructions: For each question (unless specified differently) write what would be printed (even if there are errors earlier in the code that would cause the program not to compile).

```java
import java.util.Scanner;
import java.io.*;
import java.util.Arrays;

public class Car {
```
private String make;
private String model;
private int year;
private String nickName;
private double miles;
public static Car [] carArray;

public Car (String make, String model, int year, String nickName, double miles){
    setMake(make);
    setModel(model);
    setYear(year);
    setNickName(nickName);
    setMiles(miles);
}

public String getMakeAndModel (){ return make + " " + model;
}

public void setMake (String s) {
    make = s;
}

public void setModel (String s) {
    model = s;
}

public void setYear (int i) {
    year = i;
}

public void setNickName (String s) {
    nickName = s;
}

public void setMiles (double d) {
    miles = d;
}

public int getYear () {
    return year;
}

public String getNickName() {
    return nickName;
}

public double getMiles () {
    return miles;
}

public String toString (){ String s = "Make: " + make;
    s += " Model: " + model;
    s += " Year: " + year;
    s += " Nickname: " + nickName;
    s += " Mileage: " + miles;
    return s;
}

public static void main (String [] args){
    Car c0 = new Car ("Chevy", "Camaro", 2013, "Lightning McQueen", 15000);
    Car c1 = new Car ("Ford", "F150", 1950, "Tow Mater", 200000);
    Car c2 = new Car ("Ford", "Coupe", 1936, "Doc Hudson", 150000);
Car c3 = new Car("Mack", "Flintstone", 1980, "Mack", 100000);
Car[] carsCharacters = {c0, c1, c2, c3};

//Question 1:
System.out.println(carsCharacters[2]);

//Question 2:
System.out.println(c1.getYear());

//Question 3:
for (int i = 0; i < carsCharacters.length; i++) {
    System.out.println(carsCharacters[i].getNickName());
}
}
6 General Suggestions for the Exam

6.1 Written Exam
Memorize general formats for exam, for loops, Scanners, PrintWriters, etc. I would suggest hand writing these since you’ll be doing that on the exam. Experiment, change for loops, see what happens if you don’t close PrintWriters, get a deeper understanding for the topics we’ve covered. If that isn’t a mind set that works for you, try breaking your code. Figure out what DOESN’T work, that way you won’t do it that way. Look over the first exam, find the questions you got wrong and figure out why you got them wrong. This class (and therefore this exam) is cumulative, it’s like math everything builds on each other.

6.2 Programming Exam
Re-do recitations, programming quizzes, and programming assignments that gave you a hard time with (try to do them without the internet, friends, or past assignments/recitations/quizzes) if you can’t complete them without resources you know what you need to work on. Practice writing code in Eclipse and before you run it, try and figure out what the output would be (like the Tracing portion of the written exam). Manipulate code see what works and what doesn’t. PRACTICE WRITING CODE, that is a suggestion for all of your upcoming CS classes, don’t just do the assignments make your own projects.