# Practice Exam 1 - COP 3502 Spring 2022

## **Module 4B:** Arrays

Q1. What is the output of the following program?

```
public static void main(String[] args) {
    int[] a = {11, 22, 10, 8};
    int[] b = a;
    a = new int[] {3, 13, 12, 15};
    a[3] = 2;
    for (int i : b)
        System.out.print(i + " ");
}
```

a. 11 22 10 8
b. 3 13 12 15
c. 11 22 10 2
d. 3 13 12 2
e. Error

Solution: a

Q2. Which of the following would return null?

```
public class Matrix{
    public static void main(String[] args) {
          int[][] matrix = new int[5][3];
          matrix[0] = new int[] {1, 3, 4};
          matrix[1] = new int[5];
          matrix[2] = new int[] {12, 23, 11};
    }
}
```

a. matrix[3]
b. matrix[1][2]
c. matrix[4]
d. matrix[3] and matrix[4]
e. None of the above

Solution: e

Q3. Which of the following would return null?

```
public class Matrix{
    public static void main(String[] args) {
          int[][] matrix = new int[5][];
          matrix[0] = new int[4];
          matrix[1] = new int[6];
          matrix[2] = new int[2];
          //insert array access
```

```
        }
    }
```

a. matrix[0]
b. matrix[1][1]
c. matrix[2][1]
d. matrix[3]
e. None

Solution: d

Q4. What is the output of the following program?

```
public static void main(String[] args) {
        int[][] arr = new int[4][];
        arr[0] = new int[2];
        arr[1] = arr[0];

        arr[0][0] = 3;
        arr[1][1] = 4;
        for (int i = 0; i < arr[0].length; i++) {
          System.out.print(arr[0][i] + " ");
        }
}
```

a. 3 4
b. 0 0
c. 3 0
d. null null
e. None of the above

Solution: a

## **Module 5**: Version Control and Exception

Q1. Which of the following commands will stage your entire directory and every non-empty directory inside your current directory?

a. git status all
b. git add .
c. git add all
d. git commit all

Solution: b

Q2. Which of the following statements is not correct?

a. The commit command is used to save your changes to the local repository.
b. Commits are local until they are pushed.

c. A clone operation makes a copy of a repository locally.
d. A push operation pulls down any changes made to the remote repository.

Solution: d

Q3. What is the output of the following program?

```
public static void main(String[] args) {
  try {
      int num = 4;
      num = Integer.parseInt("Interesting");
      System.out.println(num);
  }
  catch(Exception e) {
      System.out.println("Something Wrong");
  }
}
```

a. Interesting
b. Something Wrong
c. 3
d. Error

Solution: b

## **Module 6:** Class

Q1. We have a class Monkey that is defined as follows:

```
public class Monkey{
      private String name;
      private int height;
      public Monkey () {}
}
```

We define mutator methods to set the values of name and height variables of a Monkey object.

1) The return type of the mutator or setter function for the height instance variable is?
   a. int
   b. double
   c. String
   d. void

Solution: d

2) The return type of the accessor or getter function for the height instance variable is?
   a. int
   b. double
   c. String
   d. Void

Solution: a

Q2. What is the output of the following code?

```
public class Student {
	public int grade;

	public Student(int g) {
		grade = g;
	}

	public static void increaseGrades(int[] grades, Student  student,
	int grade) {
		grades[0] += 25;
		student.grade = 100;
		grade = 101;
	}

	public static void main(String[] args) {
		int[] grades = {85, 80, 90};
		Student bob = new Student(grades[1]);
		int gradeX  = grades[2];
		increaseGrades(grades, bob, gradeX);
		System.out.println(grades[0] + bob.grade + gradeX);
	}
}
```

Solution: 300

Q3. Given the following class definition,

```
public class Hero {
	String name;
	int power;
	static int heroCount = 0;

	public Hero(String name, int power) {
		this.name = name;
		this.power = power;
		heroCount++;
	}
```

```
        public static void death(int deathCount) {
              heroCount -= deathCount;
        }
}
```

What is the output of the following code snippet?
*Assume that the main method is implemented and ran properly*

```
public static void main(String args[]) {
      Hero jim = new Hero("jim", 12);
      Hero pam = new Hero("pam", 9);
      Hero andy = new Hero("andy", 4);
      jim.heroCount--;

      Hero erin = new Hero("erin", 10);
      Hero michael = new Hero("michael", 7);

      erin.death(2);

      System.out.println(Hero.heroCount);
}
```

Solution: 2

Q4. What is the output of the following program?

```
public class Apple {
    private double weight;
    public Apple(double weight) {
        this.weight = weight;
    }
    public double getWeight() {
        return weight;
    }
    public void setWeight(int _weight) {
       weight = _weight;
    }
}
public class Main {
    public static void main(String[] args) {
        Apple fuji = new Apple();
        System.out.println(fuji.getWeight());
        fuji.setWeight(3);
    }
}
```

a. 0
b. 3
c. null

d. Error

Solution: d

Q5. What is the output of the following program? (If the program is not able to be executed, please indicate "Error".)

```
public class Apple {
    private double weight;
    public Apple(double weight) {
        this.weight = weight;
    }
    public double getWeight() {
        return weight;
    }
    public void setWeight(int _weight) {
       weight = _weight;
    }
}
public class Main {
    public static void main(String[] args) {
        Apple fuji = new Apple(14);
        System.out.println(fuji.getWeight());
        fuji.setWeight(3);
        System.out.println(fuji.getWeight());
    }
}
```

Solution:

14
3

## **Module 7:** Inheritance

Q1. Given the following class names and assuming they all have default constructors:

```
class A {...}
class B extends A {...}
class C extends A {...}
class D extends C {...}
class E extends B {...}
```

Which of the following code are allowed?

a. A myClass = new A();
b. E myClass = new E();
   B nextClass = myClass;
c. C myClass = new A(); //no
d. Object myObject = new E();

e. B otherClass = new C(); //nope
f. Object myClass = new A();
   A secondClass = (A)myClass;
g. A thirdClass = new C();
   C nextClass = (C)thirdClass;

Solution:

a. **Allowed** because a class can be instantiated with itself
b. **Allowed** because subclass can be converted to super class references B => C
c. **Not Allowed** because subclasses cannot be instantiated with its super class
d. **Allowed** because every class is directly or indirectly derived from the Object class
e. **Not Allowed** because even though B and C are both derived from A, C cannot be converted to B because C is not a subclass of B
f. **Allowed** because every class is directly or indirectly derived from the Object class and **cast** can be used to convert myClass to A since myClass is an instance of A
g. **Allowed** because myClass is an instance of C and can be casted to C since myClass is an instance of C

Q2. What is the output of the above program?

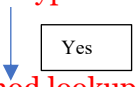
```
public class Fish {
    int weight;
    public void swim() {
        System.out.println("Splash");
    }
}
public class Salmon extends Fish {
    String home;
    public void swim() {
        System.out.println("Splish splash");
    }

    public void swim(int speed) {
        System.out.println("swimming at " + speed + " mph");
    }
}
public class Main {
    public static void main(String[] args) {
        Fish fish = new Fish();
        Salmon salmon = new Salmon();
        Fish mike = new Salmon();
        salmon.swim();
        mike.swim();
        mike.swim(5);
    }
}
```

Note: Fish class is defined in the Fish.java file
Salmon class is defined in the Salmon.java file
Main class is define in the Main.java file

Solution:

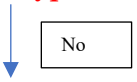
a. salmon.swim();
Since compile-time type and run-time type of salmon variable are the same. Thus, we just need to call the swim() function in the Salmon class and
**Splish splash**
Is the result

b. mike.swim();
The compile-time type of mike is Fish. The run-time type of mike is Salmon.
Step 1: Does the compile-time type "Fish" have swim() function?




Step 2: Do the dynamic method lookup: call the swim() function defined in the run-time type "Salmon". Thus,
**Splish splash**
Is the result

c. mike.swim(5);
The compile-time type of mike is Fish. The run-time type of mike is Salmon.
Step 1: Does the compile-time type "Fish" have swim(int speed) function?




**Compile Error**

Q3. Predict the output of the numbered lines in the main method. //1, //2, etc...
Write the proper output for each line.
If a line would cause an error, assume that the program would continue and write"Error".
Example answer:
1. Error
2. Output...

```
public class Student extends Person {
    public void printType() {
        System.out.println("Student");
    }
    public void printMajor() {
        System.out.println("Computer Science");
    }
}
```

```
public class Person {
    public void printType() {
        System.out.println("Normal Person");
    }

    public static void main(String[] args) {
        Person a = new Person();
        Person b = new Student();
        Object c = new Student();

        a.printType();              //1
        ((Student)a).printMajor();  //2
        b.printType();              //3
        b.printMajor();             //4
        ((Student)b).printMajor();  //5
        c.printType();              //6
        c.printMajor();             //7
        ((Person)c).printType();    //8
        ((Student)c).printType();   //9
        ((Person)c).printMajor();   //10
        ((Student)c).printMajor();  //11
        Student d = new Person();   //12
    }
}
```

Solution:

1. "Normal Person", just a normal method call
2. "Error", you can cast a superclass object to one of its subclasses.
3. "Student", **Runtime Polymorphism, Dynamic Polymorphism,** or **Dynamic Method Dispatch** all refer to the same thing. Although the reference b is a Person, the object itself is a student. When you call .printType(), Java dynamically dispatches the method to the overridden method in the Student subclass.
4. "Error", you must cast the reference to the subclass in order to call the method that only exists in the subclass.
5. "Computer Science", the superclass reference is properly casted to be of the subclass type, allowing us to call the object's printMajor method.
6. "Error", See #4
7. "Error", See#4
8. "Student" See #3 and #5
9. "Student" See #3 and #5
10. "Error", See #4
11. "Computer Science", it is casted to the subclass properly, so you can now call the method that only exists in the subclass.
12. Error, Remember: A *superclass reference variable* **can** reference an *object of a subclass* that extends the superclass. However, a subclass reference variable cannot refer to a parent class object.

Q4. What is the output of the following program?
Note: Classes are in separate files.

```
public class Grandparent {
    public Grandparent() {
        System.out.println("Grandparent constructor");
    }
}

public class Parent extends Grandparent {
    public Parent() {
        System.out.println("Parent constructor");
    }
    public Parent(int age) {
        System.out.println("Parent constructor to set up age");
    }

}

public class Child extends Parent {
    public Child() {
        System.out.println("Child constructor");
    }
    public Child(int age) {
        System.out.println("Child constructor to set up age");
    }

}

public class ConstructorDemo {
    public static void main(String[] args) {
        Child child = new Child(18);
    }
}
```

Solution:

Grandparent constructor
Parent constructor
Child constructor to set up age

Q5. What is the output of the following program?

```
public class Animal{
     public void print(){
            System.out.println("Animal printed");
     }
}
public class Dog extends Animal{
     public void print() {
```

```
            System.out.println("Dog printed");
      }
      public void print(String name){
            System.out.println("Hi, my name is " + name + " ...WOOF!");
      }
}
public class myAnimals{
      public static void main(String[] args){
            Animal doggo = new Dog();
            doggo.print("Carl");
       }
}
```

a. Animal printed
b. Dog printed
c. Hi, my name is Carl...WOOF!
d. Error

Solution:

d

Q6. Assume that the following two classes are valid

```
public class Apple {// class definition}
public class Apple extends Fruits {// class definition}
```

The following statement is declared in a main method of another class.

```
Apple fuji = new Apple();
```

Which of the followings would be the correct way to check the class membership of Apple fuji?

a. fuji is an Apple
b. fuji instanceof Apple
c. fuji is an Object
d. fuji instanceof Object
e. Apple instance of Fruits
f. Apple instance of Object
g. fuji extends Fruits

Solution:

a. Not correct, not java language
b. correct
c. Not correct, not java language
d. correct
e. Not correct, an instance of Apple class (e.g. fuji) instanceof Fruits. But a class Apple not instanceof Fruits

f. Not correct, an instance of Apple class (e.g. fuji) instanceof Object. But a class Apple is not instanceof Fruits
g. Not correct. We use the instanceof operator to check the membership.

## Coding

Q1. Write a function rotateArray(int[] arr, int d) that rotates the array arr by d elements to the left. The function header for the method is as follows:

```
public static void rotateArray(int[] arr, int d) {

}
```

Example:

Before calling rotateArray function, arr = {1, 2, 3, 4, 5}, d = 2
After calling rotateArray function, arr = {3, 4, 5, 1, 2}

**Note:** Only write the method and any helper methods, no need to create a class or create a main method.

Solution:

```
public static void rotateArray(int[] arr, int d)
{
    for (int i = 0; i < d; i++)
        leftRotatebyOne(arr);
}

public static void leftRotatebyOne(int[] arr)
{
    int temp;
    temp = arr[0];
    for (int i = 0; i < arr.length - 1; i++)
        arr[i] = arr[i + 1];
    arr[arr.length - 1] = temp;
}
```

Q2. A common statistic of interest is the longest sequence of some pattern in a list of items. For example, in football, one may be interested in the longest sequence of consecutive complete passes. Given an array of strings in which each item is either the letter "I" (for an incomplete pass) or a number (for a completed pass), output the length of the longest sequence of complete passes. Write a method that takes in an array of strings and return the longest sequence.

The function header for the method is as follows:

```
public static int longestSequence(String[] listItems) {
```

```
}
```

Example:

Input: listItems = {“4”, “15”, “9”, “I”, “30”, “2”, “I”, “20”}

Return value is 3 (because the longest sequence of complete passes is “4”, “15”, “9”).

Solution:

```
public static int longestSequence(String[] listItems) {
      // Find longest sequence of complete passes
      longestSeqLength = 0;
      currSeqLength = 0;
      for (i = 0; i < numItems; ++i) {
         // Either reached end of sequence, or no sequence yet
         if (listItems[i].equals("I")) {
            currSeqLength = 0; // Reset for the next sequence
         }
         else { // Either at start or in the middle of a sequence

            currSeqLength += 1;
            // If current sequence is longest seen so far,
            // update longest
            if (currSeqLength > longestSeqLength) {
               longestSeqLength = currSeqLength;
            }
         }
      }
      return longestSeqLength;
}
```

Q3. Write a function findDuplicates(int[] arr) that finds whether there are duplicates in the arr. If there is duplicate, return true. Otherwise, return false. The function header for the method is as follows:

```
public static boolean findDuplicates(int[] arr) {

}
```

Example:

Before calling rotateArray function, arr = {1, 2, 3, 4, 5, 3, 1}
After calling rotateArray function, return true

**Note:** Only write the method and any helper methods, no need to create a class or create a main method.

Solution:

```
public static boolean findDuplicates(int[] arr) {
          for (int i = 0; i < arr.length; i++) {
              for (int j = i + 1; j < arr.length; j++) {
                  if (arr[i] == arr[j]) {
                      return true;
                  }
              }
          }
          return false;
 }
```

Q4. Given two integers that represent the miles to drive forward and the miles to drive in reverse as user inputs, create a SimpleCar object that performs the following operations:

- Drives input number of miles forward
- Drives input number of miles in reverse
- Honks the horn
- Reports car status

The only program you need to implement is SimpleCar class. The Main class that uses SimpleCar class is provided below:

```
import java.util.Scanner;

public class LabProgram {
   public static void main(String[] args) {
      Scanner scnr = new Scanner(System.in);
      int forward;
      int reverse;
      SimpleCar car = new SimpleCar();

      forward = scnr.nextInt();
      reverse = scnr.nextInt();
      car.drive(forward);
      car.reverse(reverse);
      car.honkHorn();
      car.report();
   }
}
```

Example:

If the input is:

```
100 4
```

the output is:

```
beep beep
Car has driven: 96 miles
```

Solution:

```
// Simulates a simple car with operations to drive
// and check the odometer.
public class SimpleCar {

   // Number of miles driven
   private int miles;

   public SimpleCar(){
      miles = 0;
   }

   public void drive(int dist){
      miles = miles + dist;
   }

   public void reverse(int dist){
      miles = miles - dist;
   }

   public int getOdometer(){
      return miles;
   }

   public void honkHorn(){
      System.out.println("beep beep");
   }

   public void report(){
      System.out.println("Car has driven: " + miles + " miles");
   }
}
```

Q5. Given main(), define the Team class (in file Team.java). For class method getWinPercentage(), the formula is teamWins / (teamWins + teamLosses)

Note: Use casting to prevent integer division.

Ex: If the input is:

Ravens

13

3

where Ravens is the team's name, 13 is number of team wins, and 3 is the number of team losses,

the output is:

Congratulations, Team Ravens has a winning average!

If the input is Angels 80 82, the output is:

Team Angels has a losing average.

The only program you need to implement is Team class. The Main class WinningTeam.java that uses Team class is provided below:

```
import java.util.Scanner;

public class WinningTeam {
   public static void main(String[] args) {
      Scanner scnr = new Scanner(System.in);

      Team team = new Team();

      String name = scnr.next();
      int wins = scnr.nextInt();
      int losses = scnr.nextInt();

      team.setTeamName(name);
      team.setTeamWins(wins);
      team.setTeamLosses(losses);

      if (team.getWinPercentage() >= 0.5) {
         System.out.println("Congratulations, Team " +
team.getTeamName() + " has a winning average!");
      }
      else {
         System.out.println("Team " + team.getTeamName() +
                            " has a losing average.");
      }
   }
}
```

Solution:

```
public class Team {
   // TODO: Declare private fields - teamName, teamWins, teamLosses
   private String teamName;
   private int teamWins;
   private int teamLosses;
```

```
    // TODO: Define mutator methods -
    //       setTeamName(), setTeamWins(), setTeamLosses()
    public void setTeamName(String userName){
        teamName = userName;
    }

    public void setTeamWins(int userWins){
        teamWins = userWins;
    }

    public void setTeamLosses(int userLosses){
        teamLosses = userLosses;
    }

    // TODO: Define accessor methods -
    //       getTeamName(), getTeamWins(), getTeamLosses()
    public String getTeamName() {
        return teamName;
    }

    public int getTeamWins() {
        return teamWins;
    }

    public int getTeamLosses() {
        return teamLosses;
    }

    // TODO: Define getWinPercentage()
    public double getWinPercentage() {
        return ((double) teamWins) / (teamWins + teamLosses);
    }
}
```