# Tutorial 12

**Arrays:**

## Exercise 1:

**A.** Write a method add that receives an array of integers arr, the number of the elements in the array arr and an integer n. It then adds the integer n to the array arr if the number of elements in the array is less than its size. Method add uses another method find that checks if the integer n is in the array or not. Method add returns false if n can not be added or is already in arr.

**B.** Write a method flipCoin that receives an array of boolean flips and the number of coin flips so far. The method randomly flips a coin by calling method nextBoolean of class java.util.Random and stores the new flip in array flips if array is not full.

**C.** Write a method deleteTweet that receives your tweets, their number and a tweet that you would like to remove. The method then searches for the tweet and delete it from your twitter history. If tweet was not found, an error message is reported.

**D.** Write a method findMove that receives the history of moves made by a robot, the number of moves so far and a move. A move consists of two parts dx and dy which represent the amount of units traveled on x and y axis. The history is stored in two arrays one for each axis. The method looks up the move and returns its index in the two arrays otherwise it returns -1.

## Exercise 2:

Suppose we have the following class Customer:

```
public class Customer {
  private int id;
  private String name;
  private double totalSales;

  // Constructors, Setters, and Getters are here

  public void addSales(double price) {

  }
  public boolean equalsC(Customer c) {
    totalSales = totalSales + price;
    return (this.id == c.id &&
    this.name.equalsIgnoreCase(c.name) &&
    this.totalSales == c.totalSales);
  }
}
```

**A.** Passing an array element as an argument:
In a different class, suppose you created an array of objects of type Customer and an aarray of type double to store prices as follows:

```
Customer[] cmr = new Customer[3];
double[] prices = new double[3];

// Create objects for 1st and 2nd elements of cmr:
cmr[0] = new Customer(1, "Ahmad", 0);
cmr[1] = new Customer(2, "Saleh", 0);
prices[0] = 10.0; prices[1] = 20.0; prices[2] = 30.0;
```

1. Write a code to call the method equalsC to compare the 1st element and the 2nd element of the array cmr.

2. Write a code to call the method addSales from the 1st customer. We want to add (send) the 2nd element from the array prices.

**B.** Dealing with runtime errors:
Suppose we run this code fragment:

```
int id = cmr[2].getId();
```

What will happen?
1. Nothing, it will return the ID of the 3rd customer to be assigned to the variable id.
2. There is a compilation error.
3. There is a runtime error.

**C.** Suppose we run this code fragment:

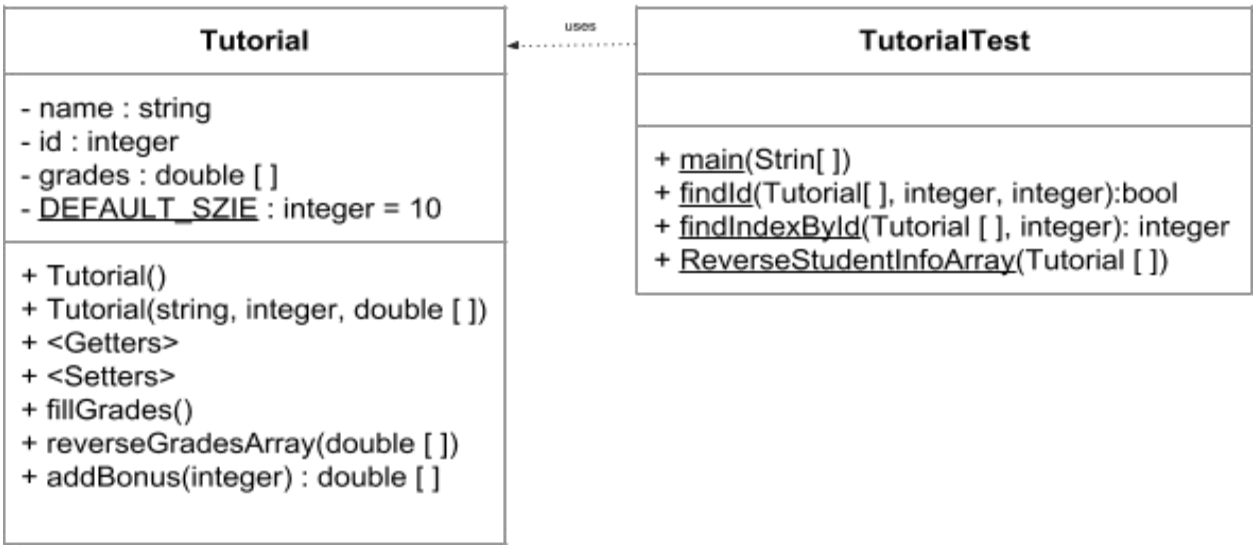
```
for (int i = 0; i <= cmr.length; i++)
  System.out.print("Name " + i + " = " + cmr[i].getName());
```

What will be the output, if any?

1. Name0 = Ahmad Name1 = Saleh Name2 = Any Name
2. There is a compilation error.
3. There is a runtime error.

## Exercise 3

Suppose we have the UML diagram for these two classes:




```
import java.util.Scanner;
public class Tutorial {
  private String name;
  private int id;
  private double[] grades;
  private static final int DEFAULT_SIZE = 10;
  public Tutorial() {
    name = "no name";
    id = -1;
    grades = new double[DEFAULT_SIZE];
  }
  public Tutorial(String newName,int newId,double[] newGrades) {
    name = newName;
    id = newId;
    grades = newGrades;
  }
  public String getName() { return name; }
  public void setName(String name) { this.name = name; }
  public int getId() { return id; }
  public void setId(int id) { this.id = id; }
  public double[] getGrades() { return grades; }
  public void setGrades(double[] grades) {this.grades = grades;}
  //The rest of the methods will be here,
  //  and they are discussed in the next pages.
}
```

- The method fillGrades() steps through the grades array of a certain student. The values are read from the user.

```
public void fillGrades(){
  Scanner kb = new Scanner(System.in);
  for(int i = 0; i < grades.length; i++)
    grades[i] = kb.nextDouble();
}
```

- The method addBonus(double bonusAmount) will add a double value to all the elements of the grades array of the current student. The value added is received using the parameter bonusAmount. After it finished adding the bonus to all the grades, the method returns the whole array to the invoker. Here is the method's code:

```
public double[] addBonus(double bonusAmount) {
  for (int i=0; i < grades.length; i++)
    grades[i] = grades[i] + bonusAmount;
}
```

- The method reverseGradesArray(double[] gradesToBeReversed) will reverse the elements of the grades array. The reversing process is done as following (grades array has n elements):
  1. The 1st element is swapped with the last element (n-1).
  2. The 2nd element is swapped with the element (n-2).
  3. The 3rd element is swapped with the element (n-3).
  4. And so on …

```
public void reverseGradesArray(double[] gradesToBeReversed) {
  double temp;
  int n = gradesToBeReversed.length;
  for (int i = 0; i < n/2; i++) {
    temp = gradesToBeReversed[i];
    gradesToBeReversed[i] = gradesToBeReversed[(n-1)-i];
    gradesToBeReversed[(n-1)-i] = temp;
  }
}
```

- Method foundId(Tutorial[] s, int id, int lastIndexReached) will:
  1. look for a student ID and return true if the student was found.
  2. help us to avoid adding an existing ID, since IDs must be unique.
  3. receive a student information array, which is an array of class Tutorial type
  4. receive the ID of the student we are looking for
  5. receive the last index reached so we look for previous elements only!

```
public static boolean foundId(Tutorial[] s, int id,
                              int lastIndexReached) {
  boolean found = false;
  for(int i = 0; i < lastIndexReached; i++)
    if(s[i].getId() == id)
      found = true;
  return found;
}
```

- Method findIndexById(Tutorial[ ] s, int id) will:
  1. look for the location of the student who's ID is id in the student information array.
  2. return the location (index) if ID is found, and -1 if not found.

```
public static int findIndexById(Tutorial[] s, int id) {
  for(int i = 0; i < s.length; i++)
    if(s[i].getId() == id)
      return i;
  return -1;
}
```

- Method reverseStudentInfoArray(Tutorial[] StudentInfoToBeReversed) will:

1. reverse the elements of the studentInfo array. Reversing is done as following:
2. The 1st element is swapped with the last element (n-1).
3. The 2nd element is swapped with the element (n-2).
4. The 3rd element is swapped with the element (n-3).
5. And so on ...

```
public static void reverseStudentInfoArray(Tutorial[]
StudentInfoToBeReversed) {
  Tutorial temp;
  int n = StudentInfoToBeReversed.length;
  for(int i = 0; i < n/2; i++) {
    temp = StudentInfoToBeReversed[i];
    StudentInfoToBeReversed[i]= StudentInfoToBeReversed[(n-1)-i];
    StudentInfoToBeReversed[(n-1)-i] = temp;
  }
}
```

- Main method: does the following:
  1. Create an array of objects studentInfo that contains students' information.
  2. Loop over the studentInfo array and create objects as elements of the array.
     I.e. adding elements to the array of objects "studentInfo":
     a. First, we ask the user to give us the ID of the current student. Then we check if the ID is unique or not.
     b. Then, we ask the user to give us the name the current student.
     c. After that, we set up the grades array of the current student.
     d. Now, we can create the object to store the current student information.
     e. Now that the studentInfo is created, we will fill the current student's grades. I.e. fill the elements of the "grades" array of the current student.
  3. Now, we will reverse a certain student's grades after we give him a bonus of 2 to all his grades.
     • Print the grades array of the selected student to assure that the bonus adding and the swapping worked properly.
  4. Then, reverse the whole studentInfo array.
     • Print the studentInfo array to assure that the swapping worked properly.

# Tutorial 12 Solutions

## Exercise 1:

**A.**
```
public int find(int[] arr, int num, int n){
    for (int i = 0; i < num; i++) {
      if (arr[i] == n) {
        return i;
      }
    }
    return -1;
  }

  public boolean add (int[] arr, int num, int n) {
    if (num < arr.length) {
      if (find(arr, num, n) == -1){
        arr[num] = n;
        num++;
        return true;
      }
      else
        System.out.println("ERROR: ELEMENT ALREADY"
    } else
    System.out.println("ERROR: ARRAY IS FULL");
    return false;
  }
```

**B.**
```
public void flipCoin (boolean[] flips, int num) {
    if (num < flips.length) {
      java.util.Random r = new java.util.Random();
      boolean newFlip = r.nextBoolean();
      flips[num] = newFlip;
      num++;
    } else
        System.out.println("ERROR: CAN NOT FLIP COIN");
  }
```

**C.**
```
public void deleteTweet(String[] tweets,int numOfTweets,String
  tweet){
    boolean found = false;
    for (int i = 0; i < numOfTweets && !found; i++) {
      if (tweets[i].equalsIgnoreCase(tweet)) {
        tweets[i] = tweets[numOfTweets];
        found = true;
      }
    }
    if (!found)
      System.out.println("ERROR: TWEET IS " + "ALREADY DELETED");
  }
```

**D.**
```
public int findMove(double[] xMoves, double[] yMoves,
                    double dx, double dy, int numMoves) {
  for (int i = 0; i < numMoves; i++)
    if ((xMoves[i] == dx) && (yMoves[i] == dy))
      return i;
   return -1;
}
```

## Exercise 2:

**A.**
```
1. if(cmr[0].equalsC(cmr[1]))
      System.out.println("They are Equal!");
// Or
   if(cmr[1].equalsC(cmr[0]))
     System.out.println("They are Equal!");
```
Note that we are sending a single element, which is sending a single object of type Customer to the method.

```
2. cmr[0].addSales(prices[1]);
```

**B.** Exception in thread "main" java.lang.NullPointerException at Tutorial13E1.main(Tutorial13E1.java:24)

- The answer is 3, we got a runtime error which is a Null Pointer Exception.
- This happened because we tried to retrieve a value of an instance variable (from within an instance method) for an object that hasn't been created!
- In other words, we created an array of objects, but we did NOT create each object of the array.
- That means we need to write the following statement before line 24 above:

```
cmr[2] = new Customer(3, "Any Name", 0);
```

**C.** Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 3 at Tutorial13E1.main(Tutorial13E1.java:29)

- The answer is C, we got a runtime error which is an Array Index Out Of Bound Exception.
- This happened because we tried to access the element cmr[3] which is not part of the array since the array has only 3 elements (indexed from 0 to 2).
- In other words, the error was caused from this operator (it should be "<"):
```
for (int i = 0; i <= cmr.length; i++)
  System.out.print("Name " + i + " = " + cmr[i].getName());
```

## Exercise 3:

```
public static void main(String[] args) {
  Scanner KB = new Scanner(System.in);
  // 1.
  System.out.print("How many students? ");
  int nstd = KB.nextInt();
  Tutorial[] studentInfo = new Tutorial[nstd];
  // 2.
  for (int i=0; i < studentInfo.length; i++) {
```

```
    // 2.a.
    do {
      System.out.print("Enter id for student "+(i+1)+":");
      int id = KB.nextInt();
      while (foundId(studentInfo, id, i));
      //2.b.
      System.out.print("Enter name for student "+(i+1)+":");
      String name = KB.next();
      // 2.c.
      System.out.print("How many grades for student "+(i+1)+"?");
      int ngr = KB.nextInt();
      double stdGrades = new double[ngr];
      // 2.d.
      studentInfo[i] = new Tutorial(name ,id, stdGrades);
      // 2.e.
      studentInfo[i].fillGrades();
    }
    // 3.
    System.out.println("Enter id for student to give bonus and
                                              reverse grades: ");
    int id = KB.nextInt();
    int index = findIndexById(studentInfo, id);
    if (id != -1) {
      double[] gr = studentInfo[index].addBonus(2);
      studentInfo[index].reverseGradesArray(gr);
      for (int i=0; i<gr.length; i++)
        System.out.println(gr[i]);
    }
    else
      System.out.println("ID not found");
    // 4.
   reverseStudentInfoArray(studentInfo);
   for (int i=0; i<studentInfo.length; i++)
     System.out.println(studentInfo[i].getId());
  }
}
```