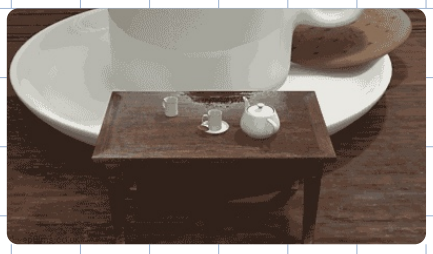
Loops, Arrays and Conditionals

kevinmiao@berkeley.edu
Kevin Miao

Material : kevin-Miao.com

- Mini lecture / Quiz Review
- Worksheet
- Attendance
- Questions



Conditionals & Loops

```
for (String i: stringarray) {
  if (x == 12) {
      break;
    } else if (x == 13) {
      continue;
    } else {
      return "Hello"; } }
```

While

```
while (i > 20) {
    i++;
    }
```

For - loops

PYTHONIAN

```
for (String i : arrString) {
    ... }
```

STANDARD

```
for (int i = 0; i < 20; i++) {
    .... }
```

Arrays

- Which one is correct?

(I)

```
values = new int[8];
```

vs

```
int[] values = new int[8];
```

(II)

```
int[] onetwo = {1, 2};
int[] onetwo = new int[] {1,2}
onetwo = {1, 2};
```

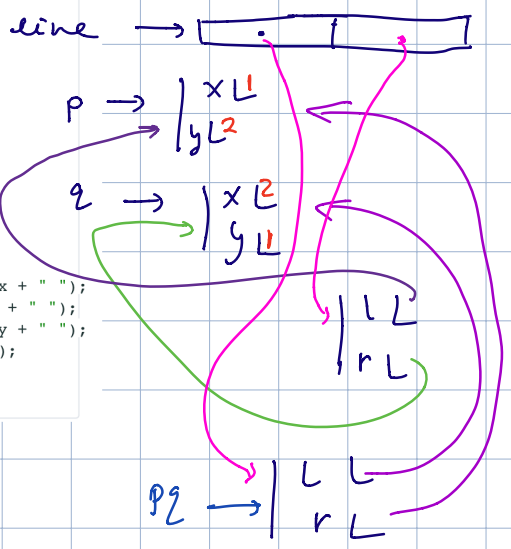
Quiz Review

3)

```
public class Point {
        public int x;
        public int y;
        //implementation
}
public class Line {
        public Point left_endpoint;
        public Point right_endpoint;
        public int slope;

        public Line(Point one, Point two) {
                this.left_endpoint = one;
                this.right_endpoint = two;
        }
        //implementation
}
```

```java
public class Test {
	public static void main(String [] ars) {
		Line[] line = new Line[2];
		Point p = new Point();
		Point q = new Point();
		Line pq = new Line(p, q);
		line[0] = pq;
		line[0].left_endpoint.x = 1;
		pq = new Line(q, p);
		line[1] = pq;
		line[0].left_endpoint.y = 2;
		line[1] = line[0];
		line[0] = pq;
		line[0].left_endpoint.x = 2;
		line[0].left_endpoint.y = 1;
		System.out.print(line[0].right_endpoint.x + " ");
		System.out.print(line[0].left_endpoint.x + " ");
		System.out.print(line[0].right_endpoint.y + " ");
		System.out.print(line[0].left_endpoint.y);
	}
}
```



4) isPrime(x)

for $1 < x < n$, is n prime?

A) $n \% x != 0$

B) $n \% x != 1$

C) $n \% x == x$

CS 61B Summer 2020 | Small Group Tutoring Section 2: Loops and Arrays | Worksheet 2

1 Read Me

Describe what each of the following methods does. You may assume that `values` contains at least one element.

```
private static boolean method1 (int[] values) {
    int k = 0;
    while (k < values.length - 1) {
        if (values[k] > values[k+1]) {
            return false;
        }
        k = k + 1;
    }
    return true;
}
```

[1, 1]

↓ True ↓ False ?

If values are in ascending order

^ not strictly

```
private static void method2 (int[] values) {
    int k = 0;
    while (k < values.length / 2) {
        int temp = values[k];
        values[k] = values[values.length - 1 - k];
        values[values.length - 1 - k] = temp;
        k = k + 1;
    }
}
```

Reverses an array

2 Flatten

Write a method flatten that takes in a 2-D int array x and returns a 1-D int array that contains all of the arrays in x concatenated together. For example, flatten({{1, 3, 7}, {}, {9}}) should return {1, 3, 7, 9}.

```
public static int[] flatten(int[][] x) {
    int newArraySize = ∅;
    for (int[] a : x) {
      nASize += a.length;
    }
    int[] newArray = new int[nASize];
    int newArrayIndex = ∅;
    for (int[] a : x) {
        for (int b : a) {
            nArray[index]= b;
            index += 1;
        }
    }
    return newArray
}
```