Wednesday February 6

Lecture 10

Compound Loop : Exercise (2.1)

```
1  System.out.println("Enter a radius value:");
2  double radius = input.nextDouble();
3  boolean isPositive = radius >= 0;
4  while (isPositive) {
5    double area = radius * radius * 3.14;
6    System.out.println("Area is " + area);
7    System.out.println("Enter a radius value:");
8    radius = input.nextDouble();
9    isPositive = radius >= 0; }
10 System.out.println("Error: negative radius value.");
```

```
1  System.out.println("Enter a radius value:");
2  double radius = input.nextDouble();
3  boolean isNegative = radius < 0;
4  while (!isNegative) {
5    double area = radius * radius * 3.14;
6    System.out.println("Area is " + area);
7    System.out.println("Enter a radius value:");
8    radius = input.nextDouble();
9    isNegative = radius < 0; }
10 System.out.println("Error: negative radius value.");
```

Test 1:
radius = -3

Test 2:
radius = 2
radius = -3

Test 3:
radius = 2
radius = 3

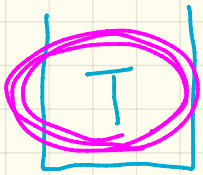
Compound Loop: Exercise (2.2)

Q: What if we delete the update at Line 9?

```
1  System.out.println("Enter a radius value:");
2  double radius = input.nextDouble();
3  boolean isPositive = radius >= 0;
4  while (isPositive) {
5      double area = radius * radius * 3.14;
6      System.out.println("Area is " + area);
7      System.out.println("Enter a radius value:");
8      radius = input.nextDouble();
9  }
10 System.out.println("Error: negative radius value.");
```

-3*-3*3.14

2*2*3.14



infinite loop.

isPositive

Test 2:

radius = 2

radius = -3

Console:

for-loop ⟷ while-loop

- To convert a `while` loop to a `for` loop, leave the initialization and update parts of the `for` loop empty.

```
while (B) {
  /* Actions */
}
```

is equivalent to:

```
for( ; B ; ) {
  /* Actions */
}
```

- To convert a `for` loop to a `while` loop, move the initialization part immediately before the `while` loop and place the update part at the end of the `while` loop body.

```
for(int i = 0 ; B ; i ++ ) {
  /* Actions */
}
```

is equivalent to:

```
int i = 0;
while (B) {
  /* Actions */
  i ++;
}
```

Stay Condition vs. Exit Condition

p q

!(p && q) == !p || !q

- When does the loop exit (i.e., stop repeating Action 1)?

```
while (p && q) { /* Action 1 */ }
```

!(p && q) : exit condition

exit condition = !(stay condition)

- When does the loop exit (i.e., stop repeating Action 2)?

```
while (p || q) { /* Action 2 */ }
```

!(p || q) : exit condition

!(p || q) == !p && !q

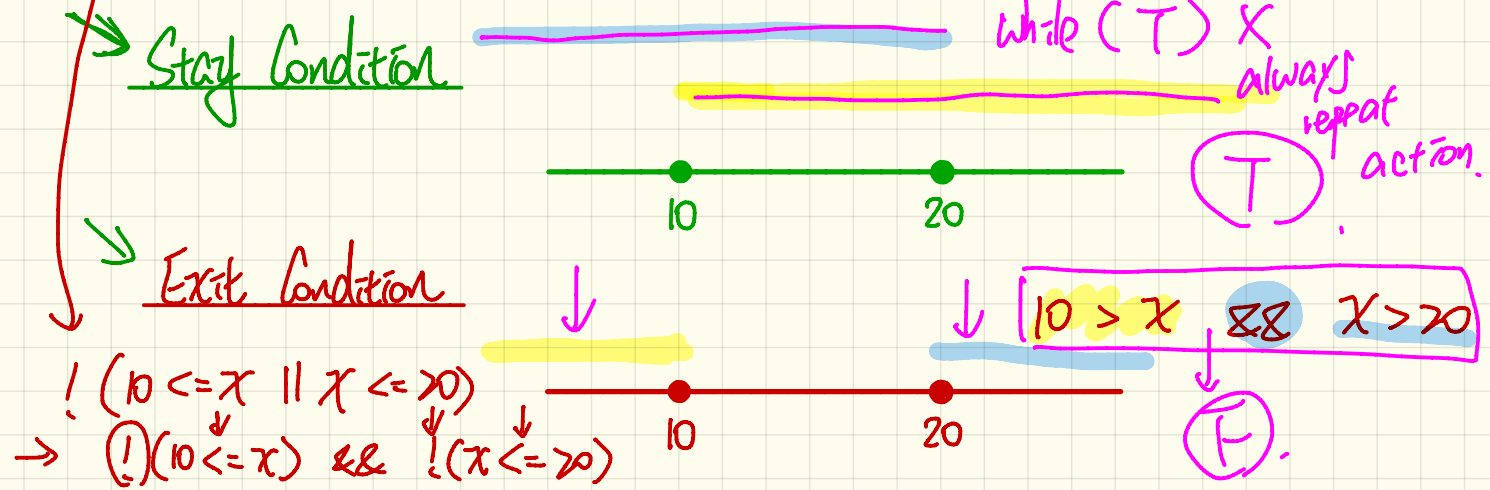
Stay Condition vs. Exit Condition: Exercise

Consider the following loop:

```
int x = input.nextInt();
while(10 <= x || x <= 20) {
  /* body of while loop */
}
```

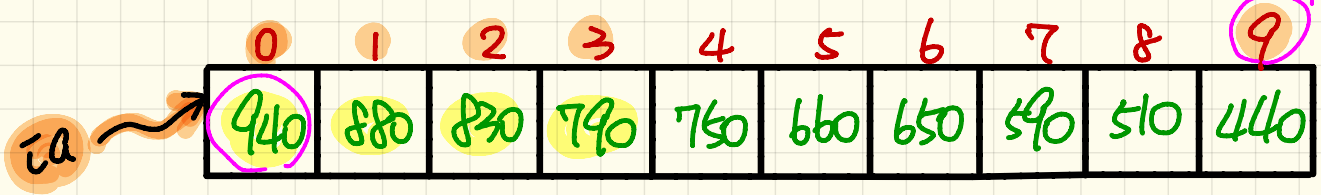
??

- It compiles, but has a logical error. Why?



Array of Integers (1)

No pattern on stored values



Declaration and Initialization: Approach 1 (Initializer)

```
int[] ia = {940, 880, 830, 790, 750, 660, 650,
            590, 510, 4440};
```

(int[] — type)

Declaration and Initialization: Approach 2 (Assignments)

```
int[] ia = new int[10];   // ← an array of int of size 10
ia[0] = 940; ia[1] = 880; ... ;
ia[9] = 4440;
```

ia.length - 1 = 9