TECHPROED

PROFESSIONAL TECHNOLOGY EDUCATION

WELCOME TO TECHPROED JAVA TUTORIAL

Testi baslatmak icin asagidaki adımları takip ediniz

Go to www.socrative.com

Click on Login

Click on Student Login

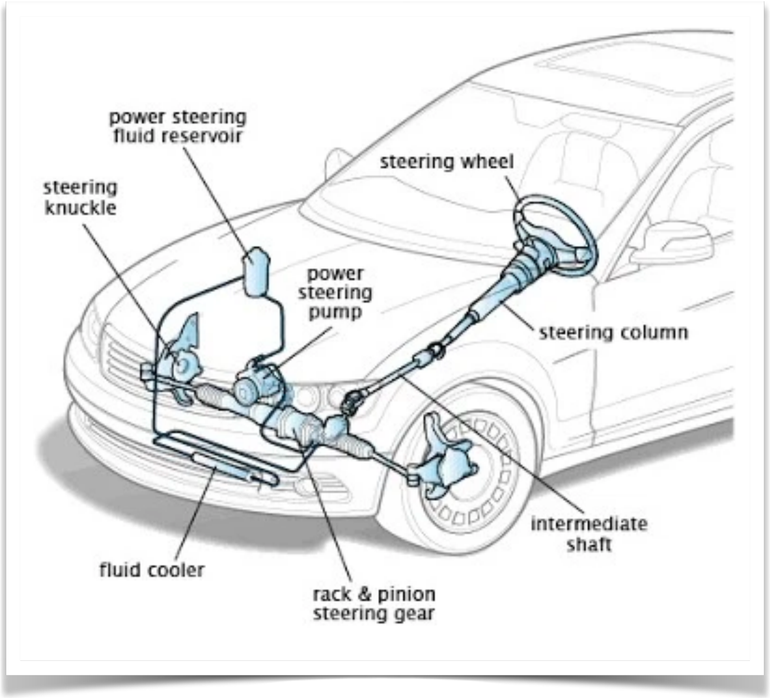
***Room Name*: A L P T E K I N 3 5 2 3**

Kayıtta kullandığınız ismi tam olarak yazınız

Time: 11 Minutes

TECHPROED
PROFESSIONAL TECHNOLOGY EDUCATION

Before Encapsulation (Data Hiding)



After Encapsulation (Data Hiding)

TECHPROED
PROFESSIONAL TECHNOLOGY EDUCATION

Encapsulation (Data Hiding)

Encapsulation data saklama (data-hiding) yöntemidir.
Eğer bir data member private ise o data'ya sadece bulunduğu class'ın icinden ulaşılabilir.

Bununla birlikte public getter ve setter methodları sayesinde private data'lar diğer class'lardan da okunabilir and update edilebilir.

Encapsulation nasıl yapılır?

Encapsulation iki adımda yapılır:

1) Data'yı*(variable, method)* private yapmalısınız.
2) public olan getter() ve setter() methodlar üretmelisiniz.
Not: getter() data'yı sadece okumamıza yarar, getter() methodu data'da degisiklik yapamaz.
Not: setter() data değerini değiştirmemize yarar

getter() method mail oluşturulur ?

```
public class CreditCards {

    private String creditCardNums = "1234567891011213";      // variable private yapıldı

    public String getCreditCardNums() {      // getter() method oluşturuldu
        return creditCardNums;
    }

}
```

Not: Eğer sadece getter method oluşturulursa data degerleri değiştirilemez sadece okunabilir. Bu tarz class'lara immutable class denir.

setter() method nasıl oluşturulur?

```
public class Account {

    private int accountBalance$ = 12345;       // variable private yapıldı
    public void setAccountBalance$ (int accountBalance$) { // setter method oluşturuldu
        this.accountBalance$ = accountBalance$;
    }

}
```

Not: Eğer sadece setter method oluşturulursa data değerleri değiştirilebilir ama okunamaz.

getter and setter methodlar birlikte de kullanilabilir

```
public class Account {

    private int accountBalance$ = 12345;      // variable private yapıldı

    public int getAccountBalance$ () {      // getter method oluşturuldu
        return accountBalance$;
    }

    public void setAccountBalance$ (int accountBalance$) {      // setter method oluşturuldu
        this.accountBalance$ = accountBalance$;
    }

}
```

Not: setter method ve getter method birlikte oluşturulursa data değerleri değiştirilebilir ve okunabilir.

Getter and setter methodlar “Java Beans” olarak da adlandırılır.
“Java Beans” leri adlandırma icin bazı kurallar vardır

1) Data type’ları boolean olan variable’ların getter metod isimleri “is” ile baslar.

```
private boolean happy = true;

public boolean isHappy() {
    return happy;
}
```

2) Data type’ları boolean olmayan variable’ların getter metod isimleri “get” ile baslar.

```
private int num = 123;

public int getNum() {
    return num;
}
```

3) Setter method isimleri her zaman “set” ile baslar

```
private String str = “Ali”;
private boolean happy = true;

public void setStr(String str) {
    this.str = str;
}

public void setHappy(boolean happy) {
    this.happy = happy;
}
```

Sorular...

1)

What is the output of the following application?

```
1: public class CompareValues {
2:   public static void main(String[] args) {
3:     int x = 0;
4:     while(x++ < 10) {}
5:     String message = x > 10 ? "Greater than" : false;
6:     System.out.println(message+","+x);
7:   }
8: }
```

A. `Greater than,10`

B. `false,10`

C. `Greater than,11`

D. `false,11`

E. The code will not compile because of line 4.

F. The code will not compile because of line 5.

2)

What is the output of the following code snippet?

```
3: java.util.List<Integer> list = new java.util.ArrayList<Integer>();
4: list.add(10);
5: list.add(14);
6: for(int x : list) {
7:   System.out.print(x + ", ");
8:   break;
9: }
```

A. `10, 14,`

B. `10, 14`

C. `10,`

D. The code will not compile because of line 7.

E. The code will not compile because of line 8.

F. The code contains an infinite loop and does not terminate.

3)

What is the output of the following code snippet?

```
3: int x = 4;
4: long y = x * 4 - x++;
5: if(y<10) System.out.println("Too Low");
6: else System.out.println("Just right");
7: else System.out.println("Too High");
```

A. `Too Low`

B. `Just Right`

C. `Too High`

D. Compiles but throws a `NullPointerException`.

E. The code will not compile because of line 6.

F. The code will not compile because of line 7.

4)

What is the output of the following code?

```
1: public class TernaryTester {
2:   public static void main(String[] args) {
3:     int x = 5;
4:     System.out.println(x > 2 ? x < 4 ? 10 : 8 : 7);
5: }}
```

A. 5

B. 4

C. 10

D. 8

E. 7

F. The code will not compile because of line 4.

5)

How many times will the following code print "Hello World"?

```
3: for(int i=0; i<10 ; ) {
4:    i = i++;
5:    System.out.println("Hello World");
6: }
```

A. 9

B. 10

C. 11

D. The code will not compile because of line 3.

E. The code will not compile because of line 5.

F. The code contains an infinite loop and does not terminate.

6)

What is the output of the following code?

```
1: public class ArithmeticSample {
2:    public static void main(String[] args) {
3:       int x = 5 * 4 % 3;
4:       System.out.println(x);
5: }}
```

A. 2

B. 3

C. 5

D. 6

E. The code will not compile because of line 3.

7)

What is the output of the following code snippet?

```
3: int x = 0;
4: String s = null;
5: if(x == s) System.out.println("Success");
6: else System.out.println("Failure");
```

A. `Success`

B. `Failure`

C. The code will not compile because of line 4.

D. The code will not compile because of line 5.

8)

What is the output of the following code snippet?

```
3: int x1 = 50, x2 = 75;
4: boolean b = x1 >= x2;
5: if(b = true) System.out.println("Success");
6: else System.out.println("Failure");
```

A. `Success`

B. `Failure`

C. The code will not compile because of line 4.

D. The code will not compile because of line 5.

9)

What is the output of the following code snippet?

```
3: do {
4:    int y = 1;
5:    System.out.print(y++ + " ");
6: } while(y <= 10);
```

A. 1 2 3 4 5 6 7 8 9

B. 1 2 3 4 5 6 7 8 9 10

C. 1 2 3 4 5 6 7 8 9 10 11

D. The code will not compile because of line 6.

E. The code contains an infinite loop and does not terminate.

10)

What is the result of the following code snippet?

```
3: int m = 9, n = 1, x = 0;
4: while(m > n) {
5:    m--;
6:    n += 2;
7:    x += m + n;
8: }
9: System.out.println(x);
```

A. 11

B. 13

C. 23

D. 36

E. 50

F. The code will not compile because of line 7.

Cevaplar...

1) F. In this example, the ternary operator has two expressions, one of them a `String` and the other a `boolean` value. The ternary operator is permitted to have expressions that don't have matching types, but the key here is the assignment to the `String` reference. The compiler knows how to assign the first expression value as a `String`, but the second `boolean` expression cannot be set as a `String`; therefore, this line will not compile.

2) C. This code does not contain any compilation errors or an infinite loop, so options D, E, and F are incorrect. The `break` statement on line 8 causes the loop to execute once and finish, so option C is the correct answer.

3) F. The code does not compile because two `else` statements cannot be chained together without additional `if-then` statements, so the correct answer is option F. Option E is incorrect as Line 6 by itself does not cause a problem, only when it is paired with Line 7. One way to fix this code so it compiles would be to add an `if-then` statement on line 6. The other solution would be to remove line 7.

4) D. As you learned in the section "Ternary Operator," although parentheses are not required, they do greatly increase code readability, such as the following equivalent statement:

```
System.out.println((x > 2) ? ((x < 4) ? 10 : 8) : 7)
```

We apply the outside ternary operator first, as it is possible the inner ternary expression may never be evaluated. Since `(x>2)` is `true`, this reduces the problem to:

```
System.out.println((x < 4) ? 10 : 8)
```

Since `x` is greater than 2, the answer is `8`, or option D in this case.

5) F. In this example, the update statement of the `for` loop is missing, which is fine as the statement is optional, so option D is incorrect. The expression inside the loop increments `i` but then assigns `i` the old value. Therefore, `i` ends the loop with the same value that it starts with: `0`. The loop will repeat infinitely, outputting the same statement over and over again because `i` remains `0` after every iteration of the loop.

6) A. The `*` and `%` have the same operator precedence, so the expression is evaluated from left-to-right. The result of `5 * 4` is `20`, and `20 % 3` is `2` (`20` divided by `3` is `18`, the remainder is `2`). The output is `2` and option A is the correct answer.

7) D. The variable `x` is an `int` and `s` is a reference to a `String` object. The two data types are incomparable because neither variable can be converted to the other variable's type. The compiler error occurs on line 5 when the comparison is attempted, so the answer is option D.

8) A. The code compiles successfully, so options C and D are incorrect. The value of `b` after line 4 is `false`. However, the `if-then` statement on line 5 contains an assignment, not a comparison. The variable `b` is assigned `true` on line 3, and the assignment operator returns `true`, so line 5 executes and displays `Success`, so the answer is option A.

9) D. The variable y is declared within the body of the `do-while` statement, so it is out of scope on line 6. Line 6 generates a compiler error, so option D is the correct answer.

10) D. Prior to the first iteration, `m = 9`, `n = 1`, and `x = 0`. After the iteration of the first loop, `m` is updated to `8`, `n` to `3`, and `x` to the sum of the new values for `m + n`, `0 + 11 = 11`. After the iteration of the second loop, `m` is updated to `7`, `n` to `5`, and `x` to the sum of the new values for `m + n`, `11 + 12 = 23`. After the iteration of the third loop, `m` is updated to `6`, `n` to `7`, and `x` to the sum of the new values for `m + n`, `23 + 13 = 36`. On the fourth iteration of the loop, `m > n` evaluates to `false`, as `6 < 7` is not `true`. The loop ends and the most recent value of `x`, `36`, is output, so the correct answer is option D.