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

Difference Between “Pass by Value” and “Pass by Reference”

When a parameter is passed by reference, the caller and the callee use the same variable for the parameter. If the callee m parameter variable, the effect is visible to the caller's variable.

When a parameter is passed by value, the caller and callee have two independent variables with the same value. If the callee modifies the parameter variable, the effect is not visible to the caller.

Java “Pass by Value” kullanır

```
public static void main(String aargs[]) {

    double price = 100;

    discountForVeteran(price);

    discountForSeniors(price);

    System.out.println(price);
}


public static void discountForVeteran(double price){
    price = price*0.80;
    System.out.println(price);
}

public static void discountForSeniors(double price){
    price = price*0.90;
    System.out.println(price);
}
```

TECHPROED
PROFESSIONAL TECHNOLOGY EDUCATION

Pass by Value and Pass by Reference Example

```
public class ReturningValues {

    public static void main(String[ ] args) {
          int number = 1;                          // 1
          String letters = "abc";                  // abc
          number(number);                          // 1
          letters = letters(letters);              // abcd

    System.out.println(number + letters); // 1abcd
   }

   public static int number(int number) {
          number++;
      return number;
  }

}
```

Local Date

```
System.out.println( LocalDate.now() );
```

Local Time

```
System.out.println( LocalTime.now() );   // 12:45:18.401
```

Note: Bu hours, minutes, seconds, ve nanoseconds gosterir

Local Date ve Time

```
System.out.println( LocalDateTime.now() );   // 2020 - 03 - 10 T 12:45:18.401
```

Note:Java T harfini date ve time ayırmak icin kullanır

Manipulating Dates

```
LocalDate date = LocalDate.now();
System.out.println(date);   // 2020 - 03 - 10
```

1)
```
date = date.plusDays(1);
System.out.println(date);   // 2020 - 03 - 11
```

OR

```
date = date.minusDays(1);
System.out.println(date);   // 2020 - 03 - 09
```

2)
```
date = date.plusMonths(2);
System.out.println(date);   // 2020 - 05 - 10
```

OR

```
date = date.minusMonths(2);
System.out.println(date);   // 2020 - 03 - 10
```

3)
```
date = date.plusYears(3);
System.out.println(date);   // 2023 - 05 - 10
```

OR

```
date = date.minusYears(3);
System.out.println(date);   // 2017 - 03 - 10
```

Manipulating Times

```
LocalTime time = LocalTime.now();
System.out.println(time);   // 12:45:18.401
```

```
time = time.plusMinutes(1);
System.out.println(time);   // 12:46:18.401
```

OR

```
time = time.minusMinutes1);
System.out.println(time);   // 12:44:18.401
```

```
time = time.plusHours(2);
System.out.println(time);   // 14:45:18.401
```

OR

```
time = time.minusHours(2);
System.out.println(time);   // 10:45:18.401
```

```
time = time.plusSeconds(3);
System.out.println(time);   // 12:45:21.401
```

OR

```
time = time.minusSeconds(3);
System.out.println(time);   // 12:45:15.401
```

```
time = time.plusNanos(5);
System.out.println(time);   // 12:45:18.406
```

OR

```
time = time.minusNanos(5);
System.out.println(time);   // 12:45:18.396
```

Formatting Date

```
DateTimeFormatter dtf = DateTimeFormatter.ofPattern(“dd-MMM-yyyy");

DateTimeFormatter dtf = DateTimeFormatter.ofPattern(“MM dd yyyy”);

LocalDate date = LocalDate.now();

dtf.format(date);
```

Formatting Time

```
DateTimeFormatter dtf = DateTimeFormatter.ofPattern(“hh:mm”);

LocalTime time = LocalTime.now();

dtf.format(time);
```

Varargs (*Variable Arguments*)

Not 1: **Vararg Parameter (*variable argument*) uzunluğu girilen argument sayısına bağlı olarak değişen arraydir.**

Not 2: **Bir Vararg Parameter method parameter'ların sonuncusu olmalıdır.**

```
public void walk( int start, int... nums ) { }    // COMPILE
```

Note 3: **Bir methodun parametrelerinden sadece bir tanesi Vararg Parameter olabilir.**

```
public void walk( int... nums, int... nums) { }      // DOES NOT COMPILE
```

TECHPROED

PROFESSIONAL TECHNOLOGY EDUCATION

Varargs Ornekleri:

```
public static void main (String[ ] args) {

    walk(1);                    // [ ]     ——→ Output is 0
    walk(2, 2);                 // [2]     ——→ Output is 1
    walk(1, 2, 3);              // [2, 3]  ——→ Output is 2

    walk(1, new int[] {4, 5}); // [4, 5]   ——→ Output is 2
}
```

```
public static void walk(int start, int... nums) {
        System.out.println(nums.length);
}
```

Access Modifiers

Access Modifier'lar bir **class**, **constructor**, **variable**, ve **method'**a erişimi kısıtlamak veya genişletmek icin kullanılır.

4 farklı Access Modifier vardır

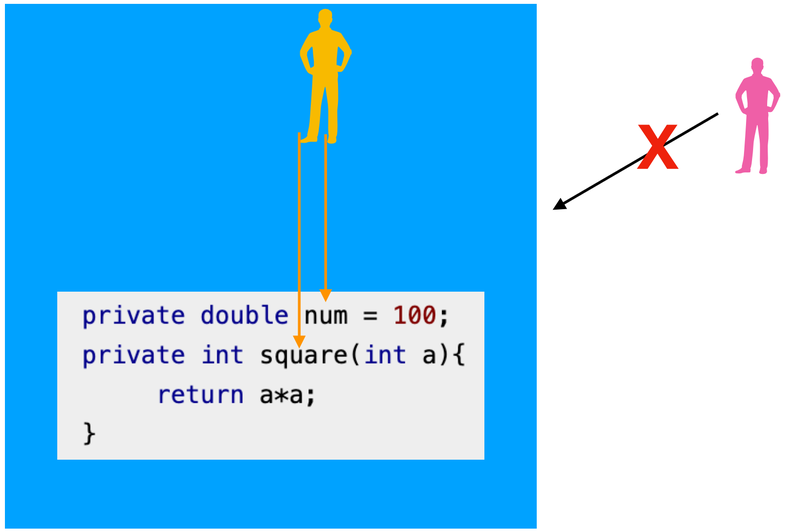
1) private (Sadece class icinde kullanılır)

2) default (Package Private) → **Eğer herhangi bir Access Modifier yazmazsak, Java Access Modifier'i default olarak kabul eder.**

3) protected

4) public

“private” Access Modifier

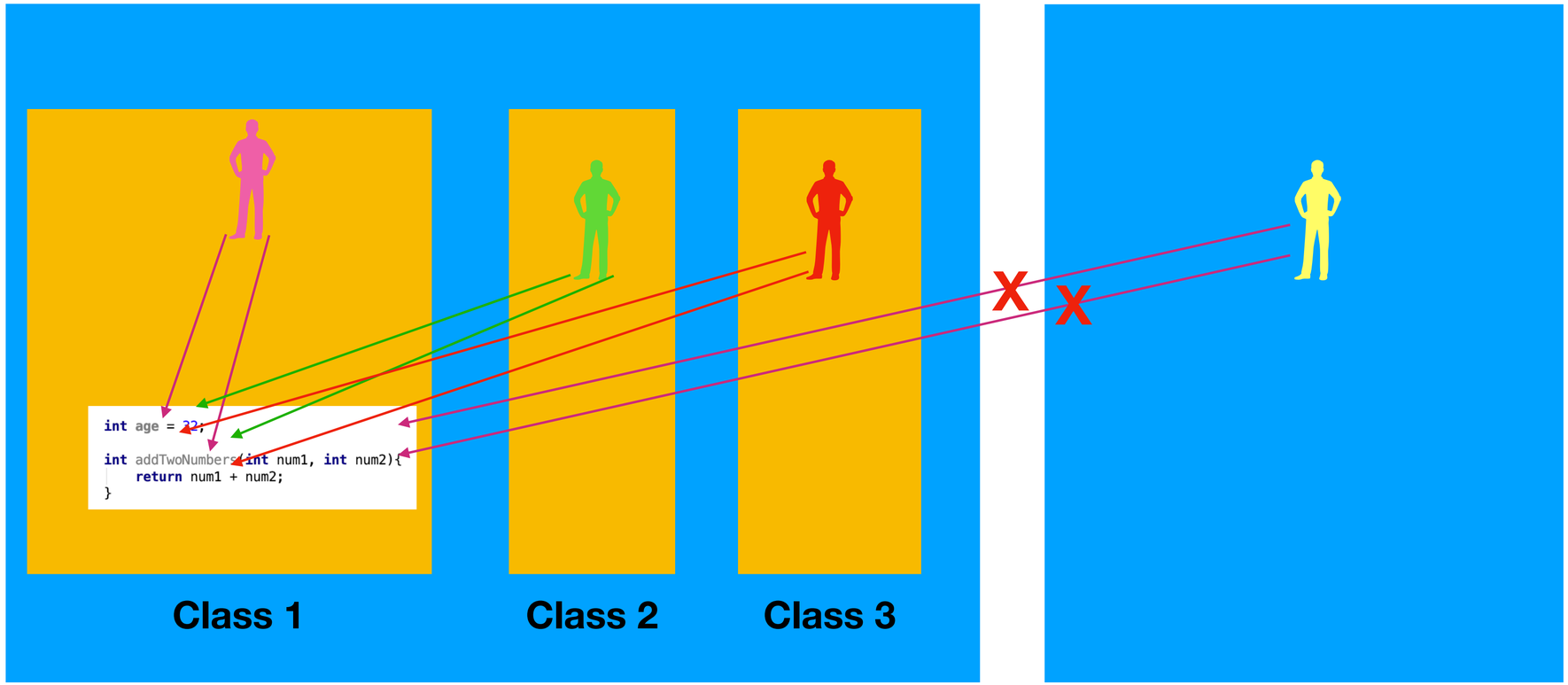


Class

Bir class için private Access Modifier kullanılamaz

TECHPROED
PROFESSIONAL TECHNOLOGY EDUCATION

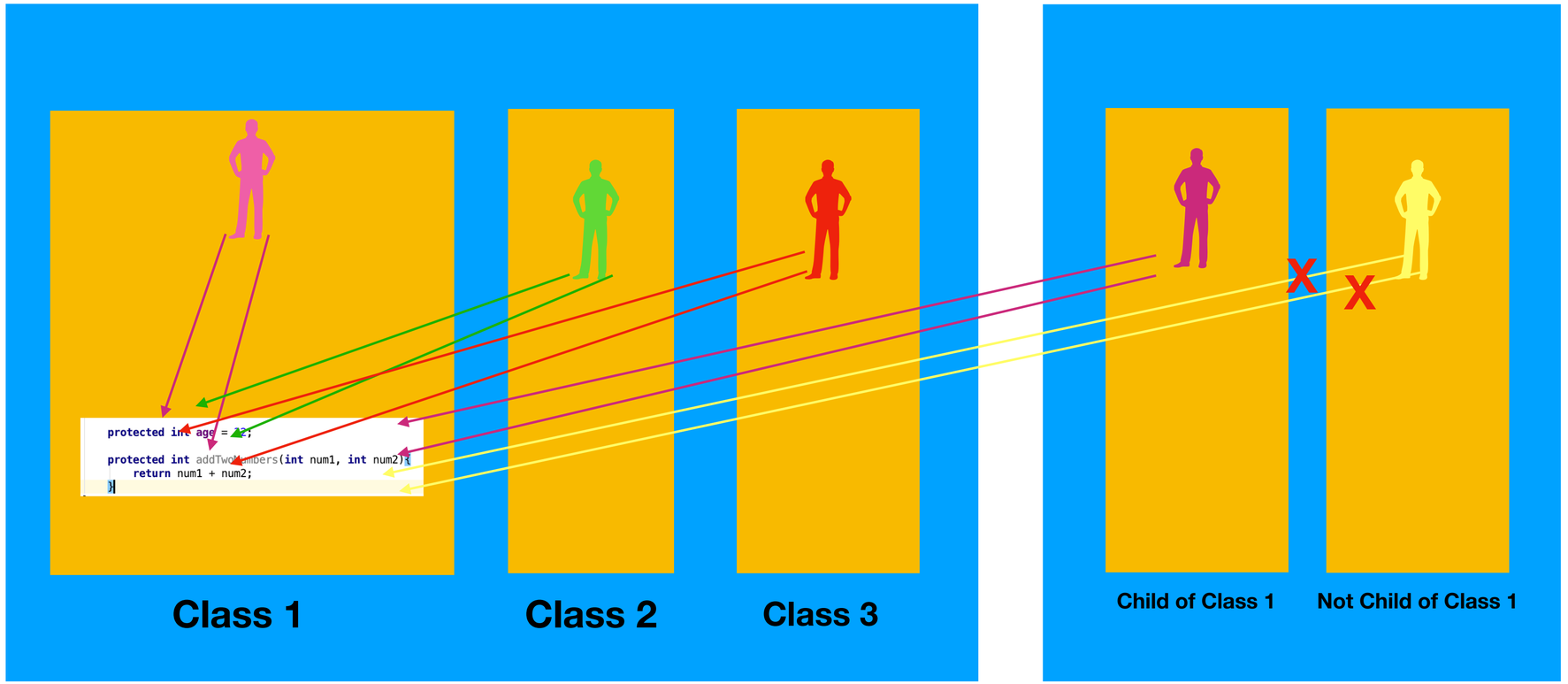
“default”(Package Private) Access Modifier



Herhangi bir Access Modifier yazılmazsa, Java onu default access modifier kabul eder.

TECHPROED
PROFESSIONAL TECHNOLOGY EDUCATION

“protected” Access Modifier



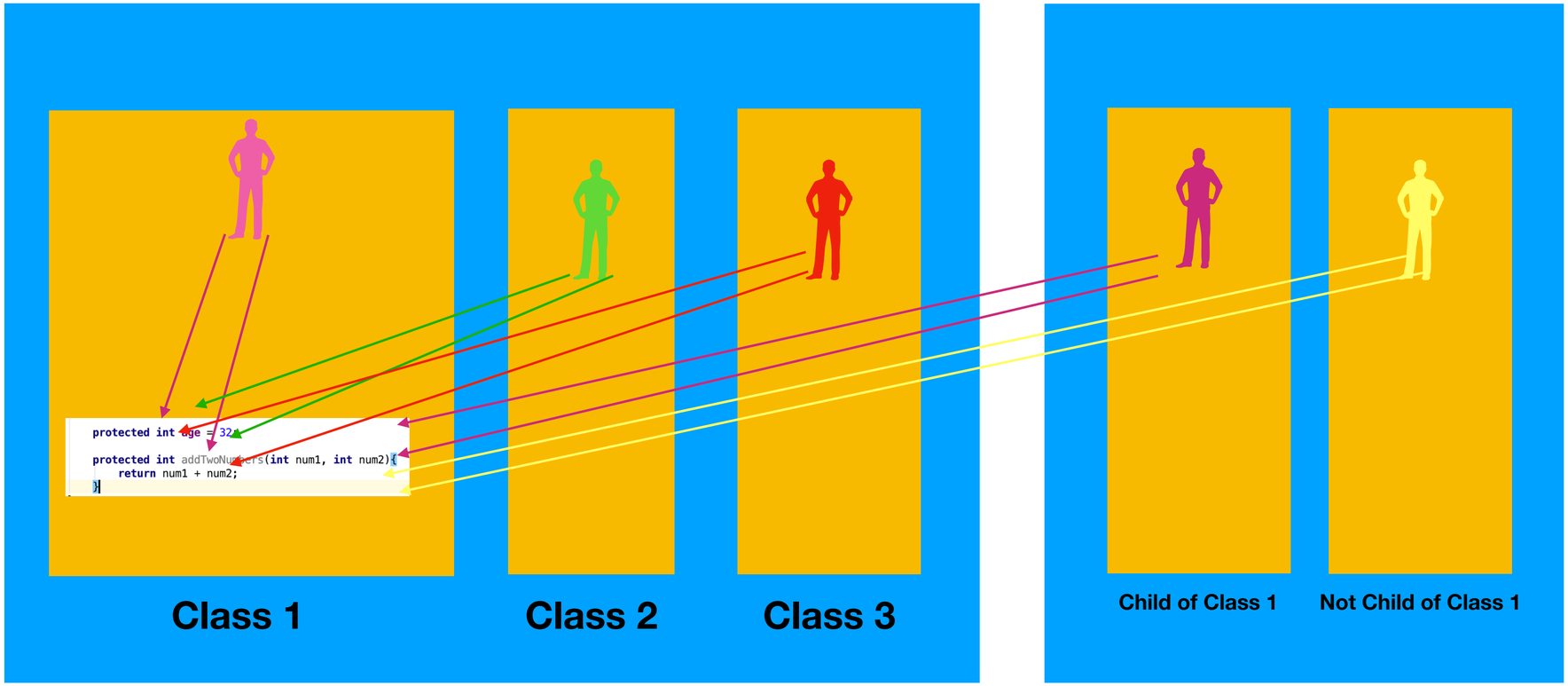
Class’lar Protected olamaz.

TECHPROED
PROFESSIONAL TECHNOLOGY EDUCATION

“public” Access Modifier

Access Modifier public secilirse, o class’a, method’a, variable’a veya constructor’a **her yerden ulaşılabilir** demektir.

“**public**” Access Modifier ulaşım için herhangi bir kısıtlama koymaz.



TECHPROED
PROFESSIONAL TECHNOLOGY EDUCATION

1)

What is the result of the following statements?

```
6:  List<String> list = new ArrayList<String>();
7:  list.add("one");
8:  list.add("two");
9:  list.add(7);
10: for(String s : list)  System.out.print(s);
```

A. onetwo

B. onetwo7

C. onetwo followed by an exception

D. Compiler error on line 9.

E. Compiler error on line 10.

2) What is the result of the following statements?

```
3: ArrayList<Integer> values = new ArrayList<>();
4: values.add(4);
5: values.add(5);
6: values.set(1, 6);
7: values.remove(0);
8: for (Integer v : values) System.out.print(v);
```

A. 4

B. 5

C. 6

D. 46

E. 45

F. An exception is thrown.

G. The code does not compile.

3) What is the result of the following?

```
int[] random = { 6, -4, 12, 0, -10 };
int x = 12;
int y = Arrays.binarySearch(random, x);
System.out.println(y);
```

A. 2

B. 4

C. 6

D. The result is undefined.

E. An exception is thrown.

F. The code does not compile.

A, B, G. Options A and
parameter declared. Opt
at all. Options C and F a
D is incorrect because tw
Option E is incorrect bec

4) Which of the following compile? (Choose all that apply)

A. `public void moreA(int... nums) {}`

B. `public void moreB(String values, int... nums) {}`

C. `public void moreC(int... nums, String values) {}`

D. `public void moreD(String... values, int... nums) {}`

E. `public void moreE(String[] values, ...int nums) {}`

F. `public void moreF(String... values, int[] nums) {}`

G. `public void moreG(String[] values, int[] nums) {}`

String Builder

1) StringBuilder sb1 = new StringBuilder(); ===> Boş bir StringBuilder oluşturur

2) StringBuilder sb2 = new StringBuilder("animal"); ===> Belli bir degeri olan StringBuilder oluşturur

3) StringBuilder sb3 = new StringBuilder(5); ===> Ilk uzunluğu tahmin edilen bir StringBuilder oluşturur.

Note: StringBuffer, StringBuilder'a benzer. StringBuilder, StringBuffer'dan hızlıdır.
Multi-thread icin StringBuffer kullanılır.

```
StringBuilder sb = new StringBuilder(5);
```

0	1	2	3	4

```
sb.append("anim");
```

a	n	i	m	
0	1	2	3	4

```
sb.append("als");
```

a	n	i	m	a	l	s		
0	1	2	3	4	5	6	7	...

StringBuilder Methods

```
StringBuilder sb = new StringBuilder("animals");
```

1)
```
String sub1 = sb.substring( 3 );
System.out.println( sub1 );
```

2)
```
String sub2 = sb.substring( 2, 5 );
System.out.println( sub2 );
```

3)
```
sb.indexOf("n")
```

4)
```
int lng = sb.length();
System.out.println( lng );
```

5)
```
char ch = sb.charAt(6);
System.out.println( ch );
```

6) StringBuilder sb1 = new StringBuilder();

```
sb1.append("A").append("b");
System.out.println(sb1);      // Ab
StringBuilder sb2 = new StringBuilder().append("A");
sb2.append("b").append("c");
System.out.println(sb2);      // Abc
```

7) StringBuilder sb1 = new StringBuilder("animal");

```
sb1.insert(0,"X");
System.out.println(sb1);      // Xanimal
sb1.insert(7,"X");
System.out.println(sb1);      // XanimalX
sb1.insert(4,"X");
System.out.println(sb1);      // XaniXmalX
```

8) StringBuilder sb1 = new StringBuilder("abcdef");

```
sb1.delete(1, 3);
System.out.println(sb1);      // adef
```

9) StringBuilder sb2 = new StringBuilder("abcdef");

```
sb2.deleteCharAt(2);
System.out.println(sb2);      // abed

sb2.deleteCharAt(4); // throws an exception
```

10) StringBuilder sb1 = new StringBuilder("abc");

```
sb1.reverse();
System.out.println(sb1);      // cba
```

11) StringBuilder sb2 = new StringBuilder("abc");

```
sb2.toString();
System.out.println(sb2);      // abc
```

Note: StringBuilder was added to Java in Java 5.
If you come across older code, you will see StringBuffer used for this purpose.
StringBuffer does the same thing but more slowly; therefore, use StringBuilder

Sorular…

1) What is the result of the following code?

```
7: StringBuilder sb = new StringBuilder();
8: sb.append("aaa").insert(1, "bb").insert(4, "ccc");
9: System.out.println(sb);
```

A. abbaaccc

B. abbaccca

C. bbaaaccc

D. bbaaccca

E. An exception is thrown.

F. The code does not compile.

2) What is the result of the following code?

```
2: String s1 = "java";
3: StringBuilder s2 = new StringBuilder("java");
4: if (s1 == s2)
5:  System.out.print("1");
6: if (s1.equals(s2))
7:  System.out.print("2");
```

A. 1

B. 2

C. 12

D. No output is printed.

E. An exception is thrown.

F. The code does not compile.

3) Which are the results of the following code? (Choose all that apply)

```
String numbers = "012345678";
System.out.println(numbers.substring(1, 3));
System.out.println(numbers.substring(7, 7));
System.out.println(numbers.substring(7));
```

A. 12

B. 123

C. 7

D. 78

E. A blank line.

F. An exception is thrown.

G. The code does not compile.

4) What is the result of the following code?

```
4: int total = 0;
5: StringBuilder letters = new StringBuilder("abcdefg");
6: total += letters.substring(1, 2).length();
7: total += letters.substring(6, 6).length();
8: total += letters.substring(6, 5).length();
9: System.out.println(total);
```

A. 1

B. 2

C. 3

D. 7

E. An exception is thrown.

F. The code does not compile.

5) . Which are true of the following code? (Choose all that apply)

```
1:  public class Rope {
2:    public static void swing() {
3:      System.out.print("swing ");
4:    }
5:    public void climb() {
6:      System.out.println("climb ");
7:    }
8:    public static void play() {
9:      swing();
10:     climb();
11:   }
12:   public static void main(String[] args) {
13:     Rope rope = new Rope();
14:     rope.play();
15:     Rope rope2 = null;
16:     rope2.play();
17:   }
18: }
```

A. The code compiles as is.
B. There is exactly one compiler error in the code.
C. There are exactly two compiler errors in the code.
D. If the lines with compiler errors are removed, the output is `climb climb`.
E. If the lines with compiler errors are removed, the output is `swing swing`.
F. If the lines with compile errors are removed, the code throws a `NullPointerException`.

B, E. Line 10 does not compile because static methods are not allowed to call instance methods. Even though we are calling `play()` as if it were an instance method and an instance exists, Java knows `play()` is really a static method and treats it as such. If line 10 is removed, the code works. It does not throw a `NullPointerException` on line 16 because `play()` is a static method. Java looks at the type of the reference for `rope2` and translates the call to `Rope.play()`.

6) What is the result of the following program?

```
1: public class Squares {
2:   public static long square(int x) {
3:     long y = x * (long) x;
4:     x = -1;
5:     return y;
6:   }
7:   public static void main(String[] args) {
8:     int value = 9;
9:     long result = square(value);
10:     System.out.println(value);
11:   } }
```

A. -1

B. 9

C. 81

D. Compiler error on line 9.

E. Compiler error on a different line.

B. Since Java is pass-by-value and the variable on line 8 never gets reassigned, it stays as 9. In the method `square`, *x* starts as 9. *y* becomes 81 and then *x* gets set to –1. Line 9 does set *result* to 81. However, we are printing out *value* and that is still 9.