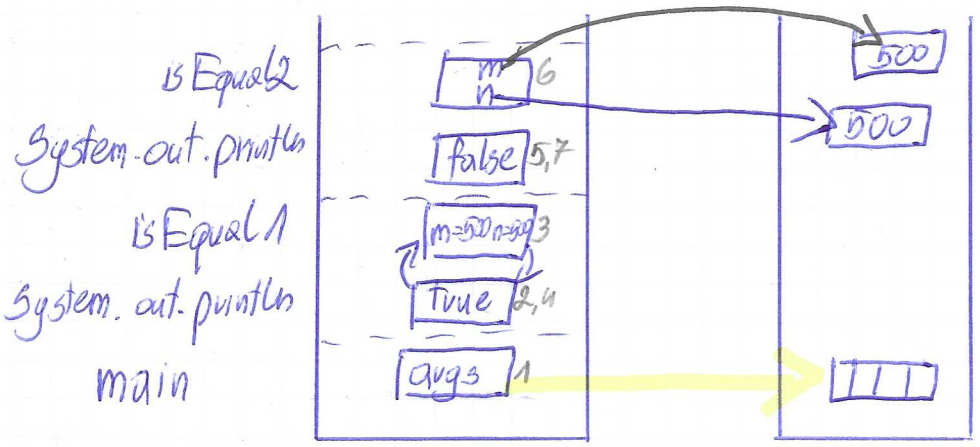
3. Wydruk: true, false



STOS
(rekordy aktywacji
→ pamięć automatyczna)

STERTA
(pamięć dynamiczna
Garbage collector)

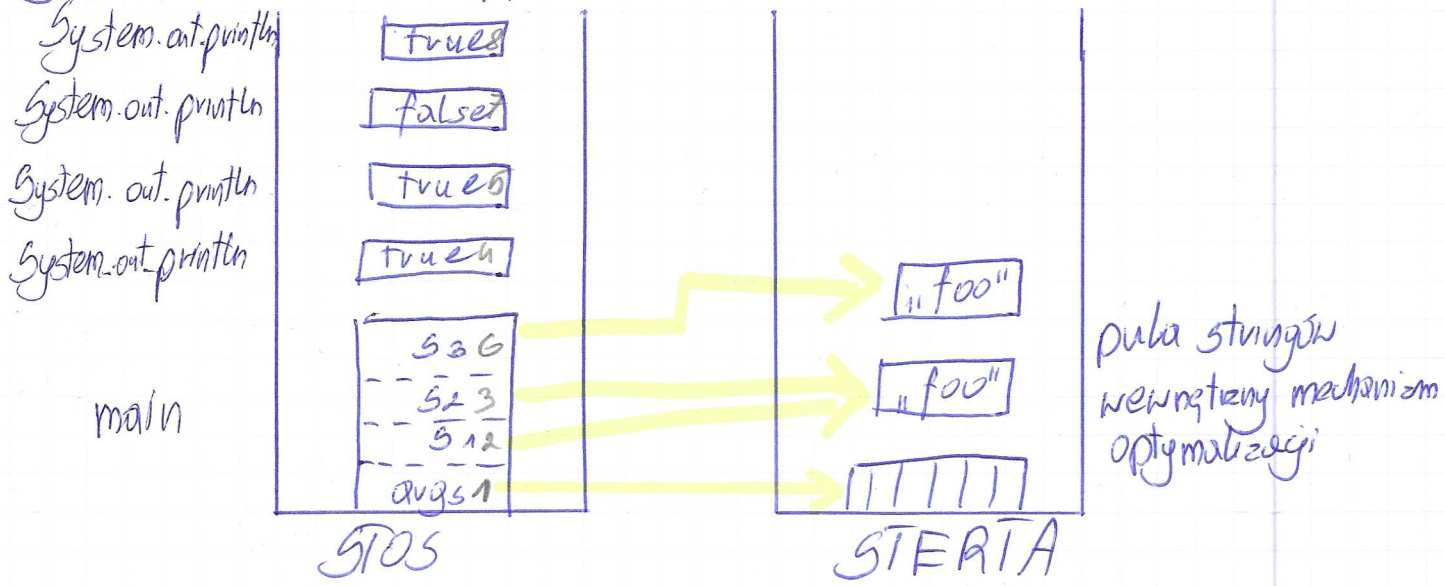
przy wywołaniu funkcji, wejściu bloku na stos

Początkowo typy proste int porównywane fizycznie (ta sama komórka) zwróci true. Przy porównaniu typów obiektowych (referencyjnych) Integerach m i n, referencji do obiektów różnych równość referencji == zwróci false.
(adresów na stercie)

3. Co i **dlaczego** wydrukuje poniższy program w Javie?

```
public class IsEqual{
  static boolean isEqual1(int m, int n){return m == n;}
  static boolean isEqual2(Integer m, Integer n){return m == n;}
  public static void main(String[] args){
    System.out.println(isEqual1(500,500));
    System.out.println(isEqual2(500,500));
  }
}
```

4. Wydruk: true, true, false, true



Klasa String nie potrzebuje new. s_1, s_2 referencje tego samego obiektu String „foo". s_3 poprzez new tworzy się zupełnie nowy obiekt o tej samej wartości, co poprzedni.

$s_1 == s_2$, porównujemy czy odnoszą się do tego samego obiektu, tak

s_1.equals(s_2), porównujemy, czy mają te same wartości, tak

$s_1 == s_3$, ten sam obiekt? nie

s_1.equals(s_3), te same wartości? tak

4. Co i dlaczego wydrukuje poniższy program w Javie?

```
public class Porównanie{
   public static void main(String[] args){
      String s1 = "foo";
      String s2 = "foo";
      System.out.println(s1 == s2);
      System.out.println(s1.equals(s2));
      String s3 = new String("foo");
      System.out.println(s1 == s3);
      System.out.println(s1.equals(s3));
   }
}
```