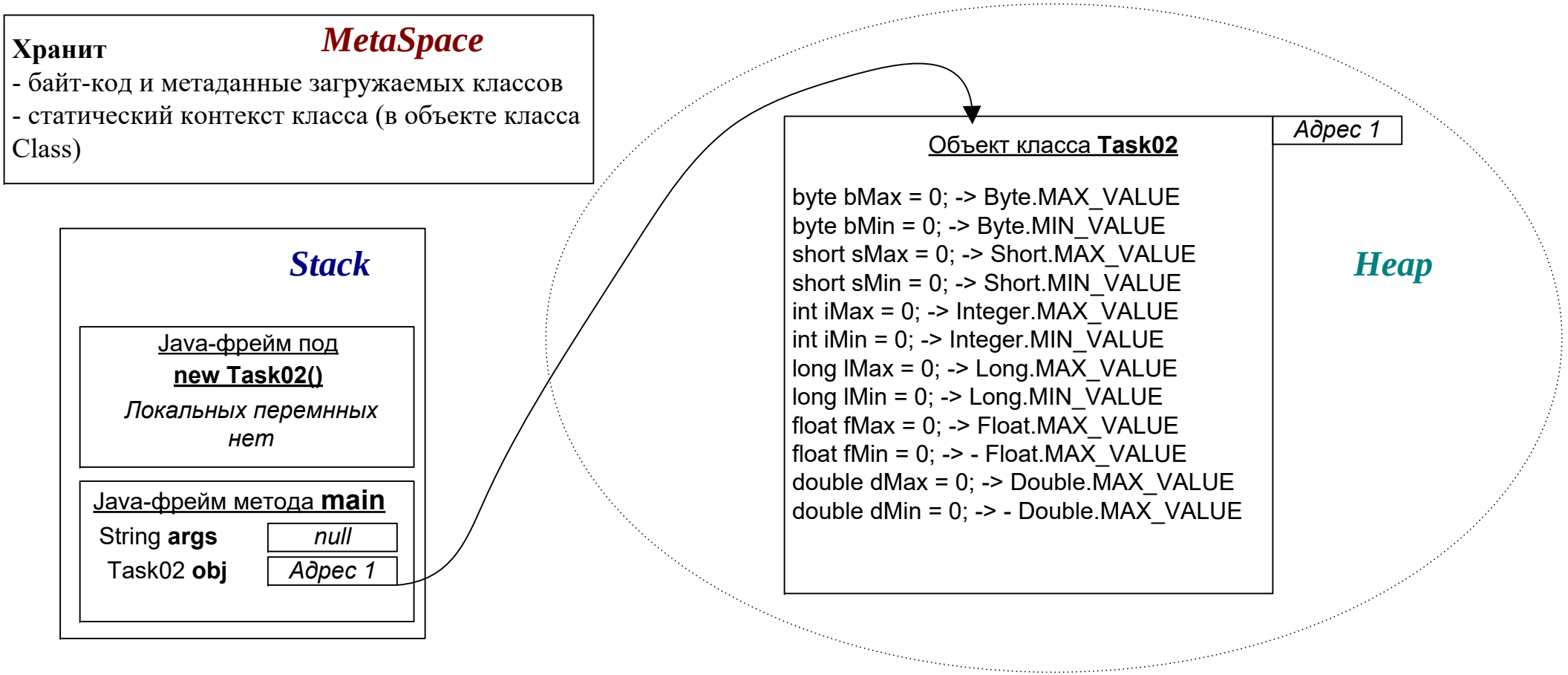
```
public static void main(String[] args) {
        Task02 obj = new Task02 (); // 0
```

0

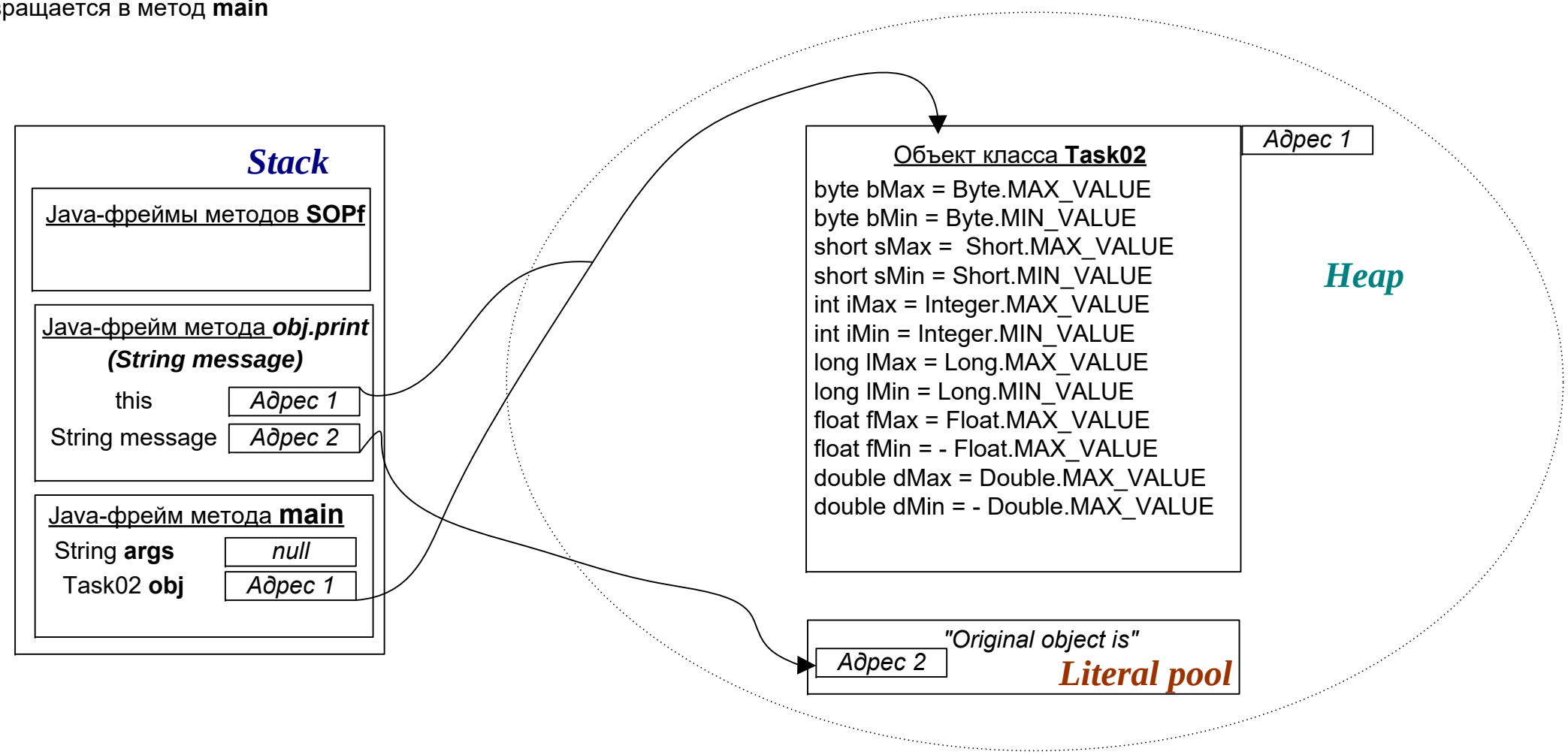
1) В стековой памяти выделятся фрейм под исполнение метода **main**
2) Во фрейме выделяется память под 2 ссылочные переменные: ***args*** (если параметров командной строки не было указано, то инициализируется null) и ***obj***
3) Исполняется оператор new, который на основании сведений о классе из ***Metaspace*** очищает в ***Heap*** требуемый для экземпляра класса объем памяти под объект класса Task02, тем самым записывая туда нулевые значения (что-то такое на памяти осталось из литературы).
4) После чего осуществляется вызов конструктора класса Task02, который производит инициализацию полей класса Task02 требуемыми в соответствии с условиями задачи значениями.



```
public static void main(String[] args) {
        ….
    obj.print("Original object is"); // 1
```

1

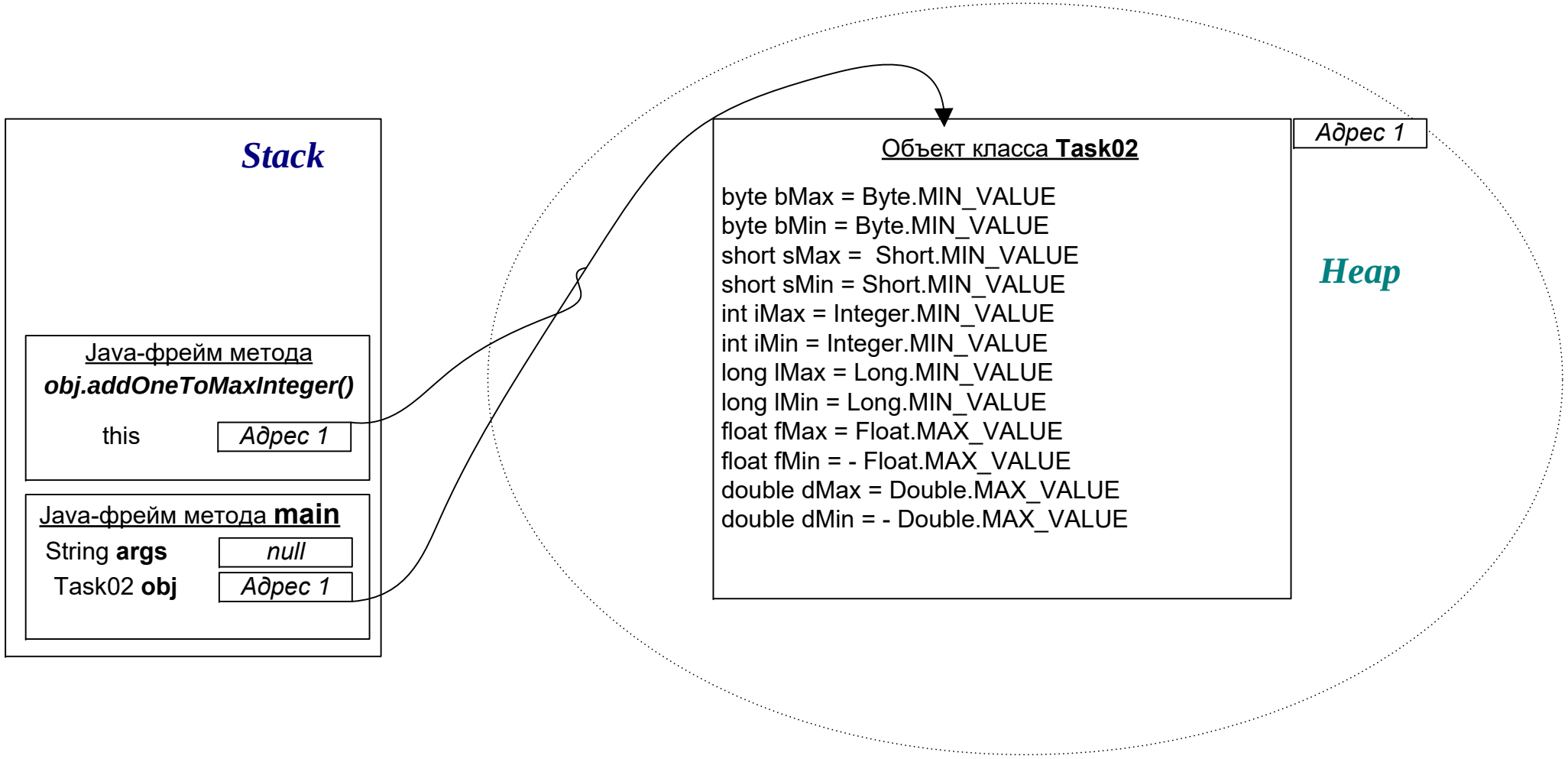
1) Java-фрейм, выделенный под исполнение **new Task02()**, удаляется из стека, исполнение программы возвращается в метод **main**
2) Исполнение переходит к методу **obj.print (String message)**, перед вызовом которого в пуле литералов, расположенном в Heap (начиная с Java 7), создается строковый литерал "Original object is", ссылка на который и будет передана в метод obj,print (String message).
3) Во фрейме метода **obj.print (String message)** выделяется память под 1 ссылочную переменную ***message***, инициализированную адресом литерала "Original object is", и под ссылочную переменную **this**, указывающую на объект, которым был инициирован вызов данного метода.
3) Затем последовательно в стеке создается фрейм для очередного System.out.printf, он исполняется, фрейм удаляется, создается новый фрейм под следующий System.out.printf и т.д.
4) После исполнения последнего System.out.printf из стека удаляется фрейм метода **obj,print (String message)** и управление возвращается в метод **main**



```
public static void main(String[] args) {
        ….
    obj.addOneToMaxInteger(); // 2
```

2

1) Исполнение переходит к методу ***obj.addOneToMaxInteger()***, в стеке формируется фрейм, в котором выделяется память под ссылочную переменную **this**, указывающую на объект, которым был инициирован вызов данного метода.
2) В методе значения «максимальных переменных» целочисленного типа увеличиваются на 1 и «двоичная логика», проскладывая его с каждым битом числа справа налево, переводит этот единичный бит в знаковую позицию числа (крайнюю левую), тем самым получаю максимальное отрицательное значение данного типа.
3) После исполнения последнего оператора инкремента из стека удаляется фрейм данного метода и управление возвращается в метод **main**



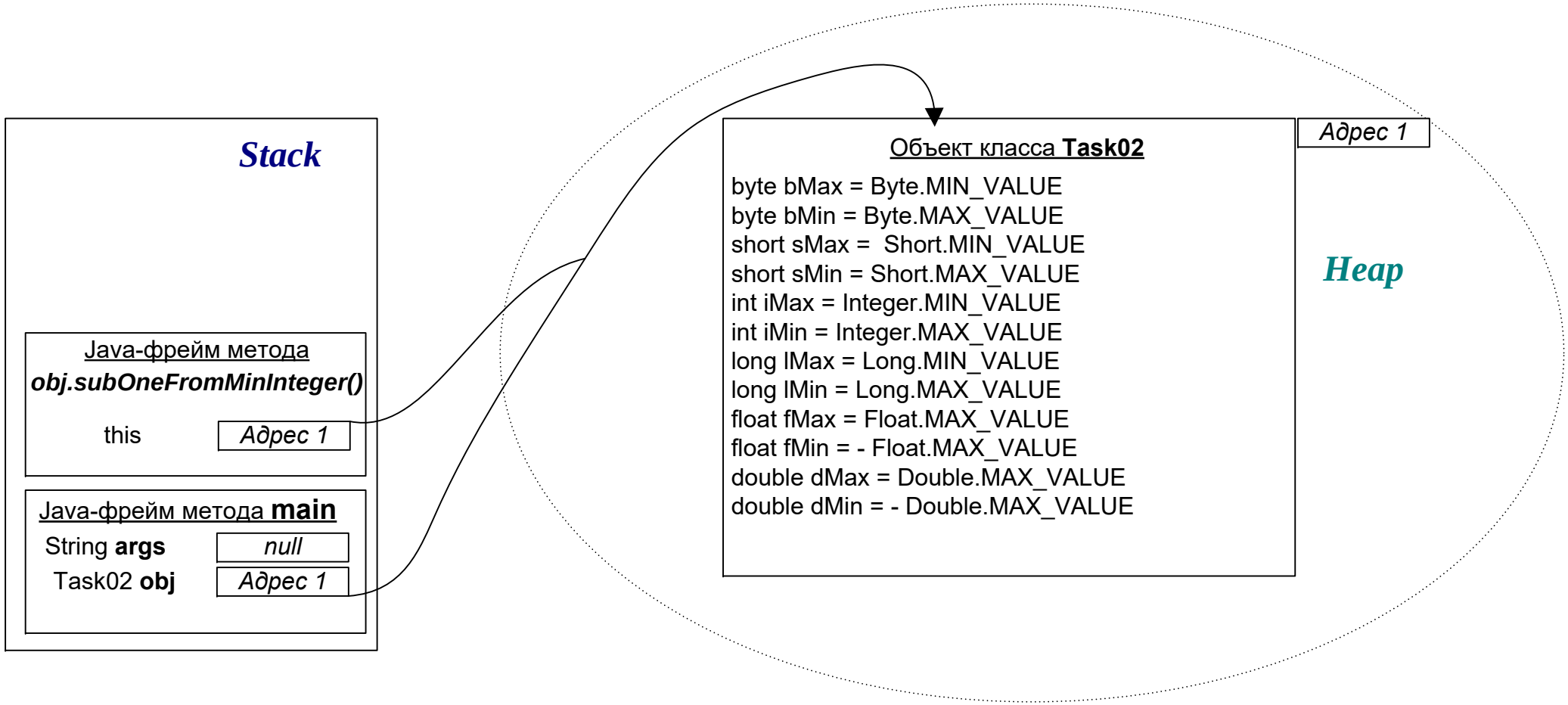
```
public static void main(String[] args) {
        ….
  obj.print("Object with one added to MAX-integer fields"); // 3
```

1) Исполнение данного метода аналогично исполнению метода «1»

```
public static void main(String[] args) {
        ….
    obj.subOneFromMinInteger(); // 4
```

4

1) Вся «подкапотная» работа аналогична методу «2»
2) Отличия долько в том, что в методе значения «минимальных переменных» целочисленного типа уменьшаются на 1 и «двоичная логика», проотнимая его от каждого нулевого бита числа справа налево, переводит все биты в единицы, а знаковую позицию числа (крайнюю левую) - обнуляет, тем самым получаю максимальное положительное значение данного типа.
3) После исполнения последнего оператора декремента из стека удаляется фрейм данного метода и управление возвращается в метод **main**



```
public static void main(String[] args) {
        ….
    obj.print("Object with one subtracted from MIN-integer fields"); // 5
```

1) Исполнение данного метода аналогично исполнению методов «1» и «3».