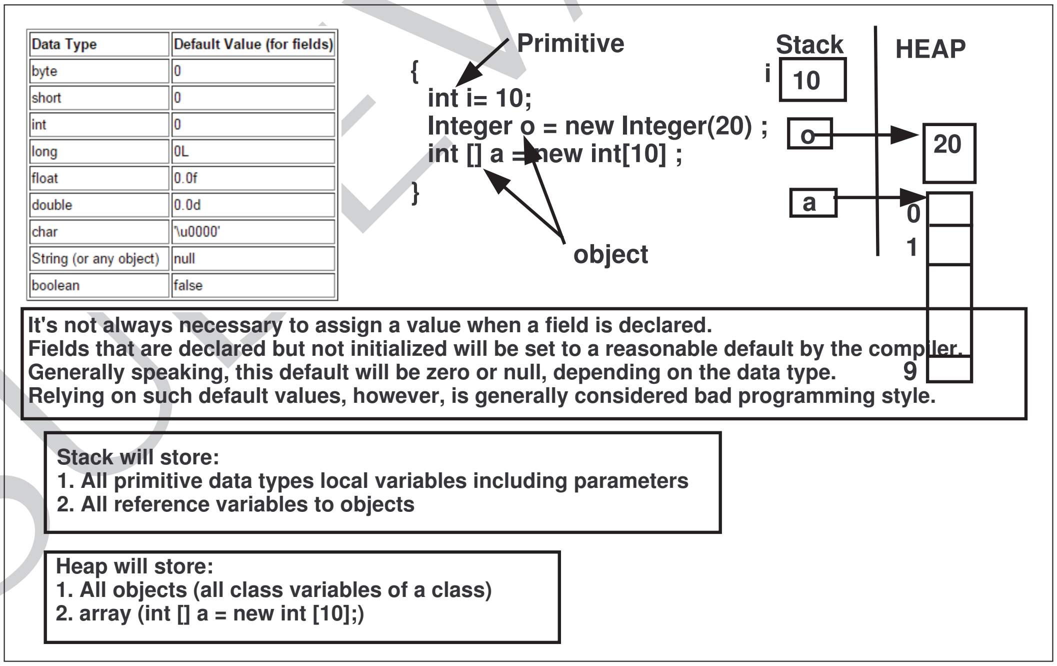
Chapter 1

Java

1.1 Data types supported by Java



Data Type	Default Value (for fields)
byte	0
short	0
int	0
long	0L
float	0.0f
double	0.0d
char	'\u0000'
String (or any object)	null
boolean	false

```
{
  int i= 10;
  Integer o = new Integer(20) ;
  int [] a = new int[10] ;
}
```

It's not always necessary to assign a value when a field is declared.
Fields that are declared but not initialized will be set to a reasonable default by the compiler.
Generally speaking, this default will be zero or null, depending on the data type.
Relying on such default values, however, is generally considered bad programming style.

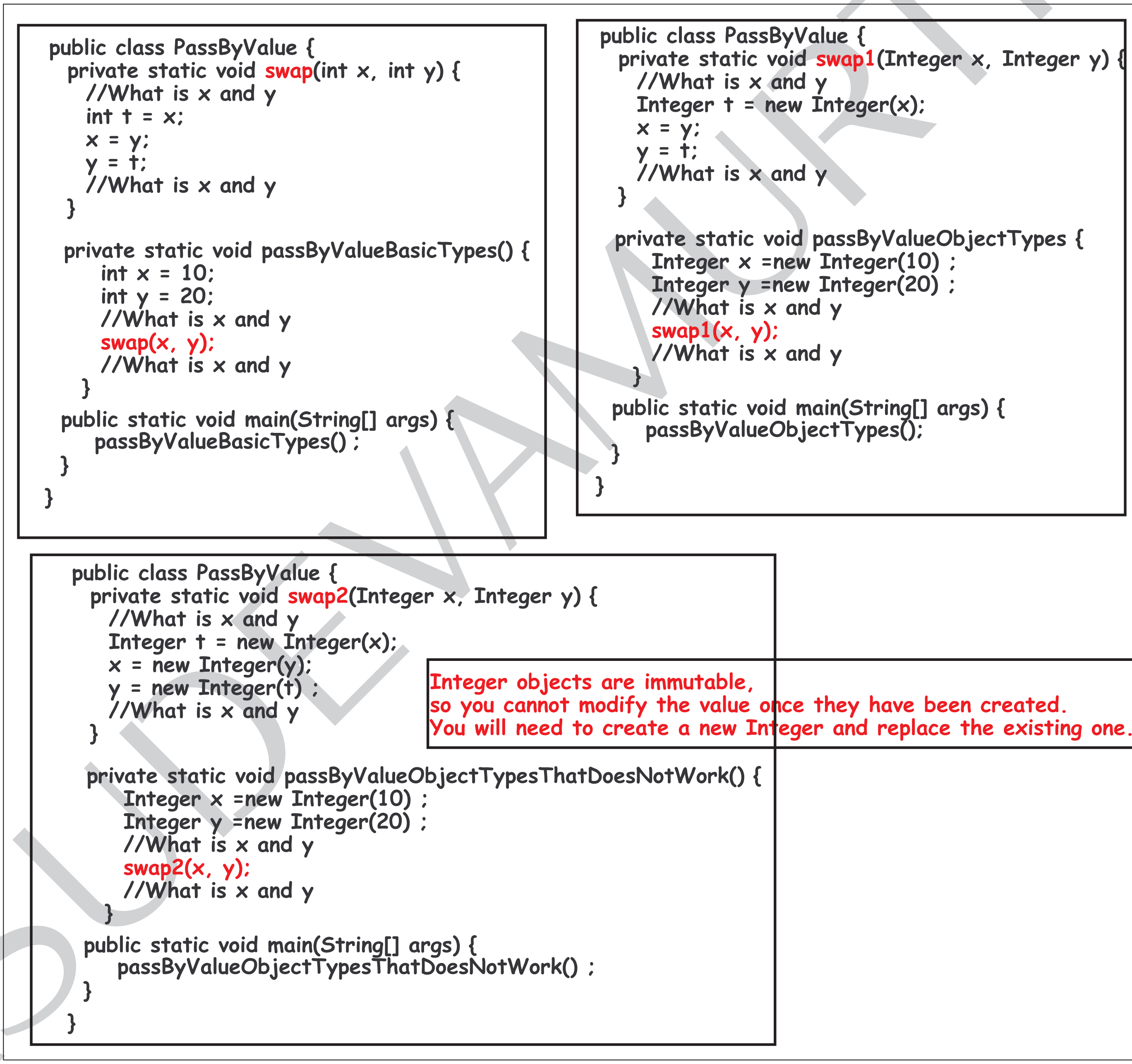
Stack will store:
1. All primitive data types local variables including parameters
2. All reference variables to objects

Heap will store:
1. All objects (all class variables of a class)
2. array (int [] a = new int [10];)

Figure 1.1: Java data types

1.2 Pass by value

1.2.1 Pass by value - Primitive types



```
public class PassByValue {
  private static void swap(int x, int y) {
    //What is x and y
    int t = x;
    x = y;
    y = t;
    //What is x and y
  }

 private static void passByValueBasicTypes() {
     int x = 10;
     int y = 20;
     //What is x and y
     swap(x, y);
     //What is x and y
   }
 public static void main(String[] args) {
     passByValueBasicTypes() ;
 }
}
```

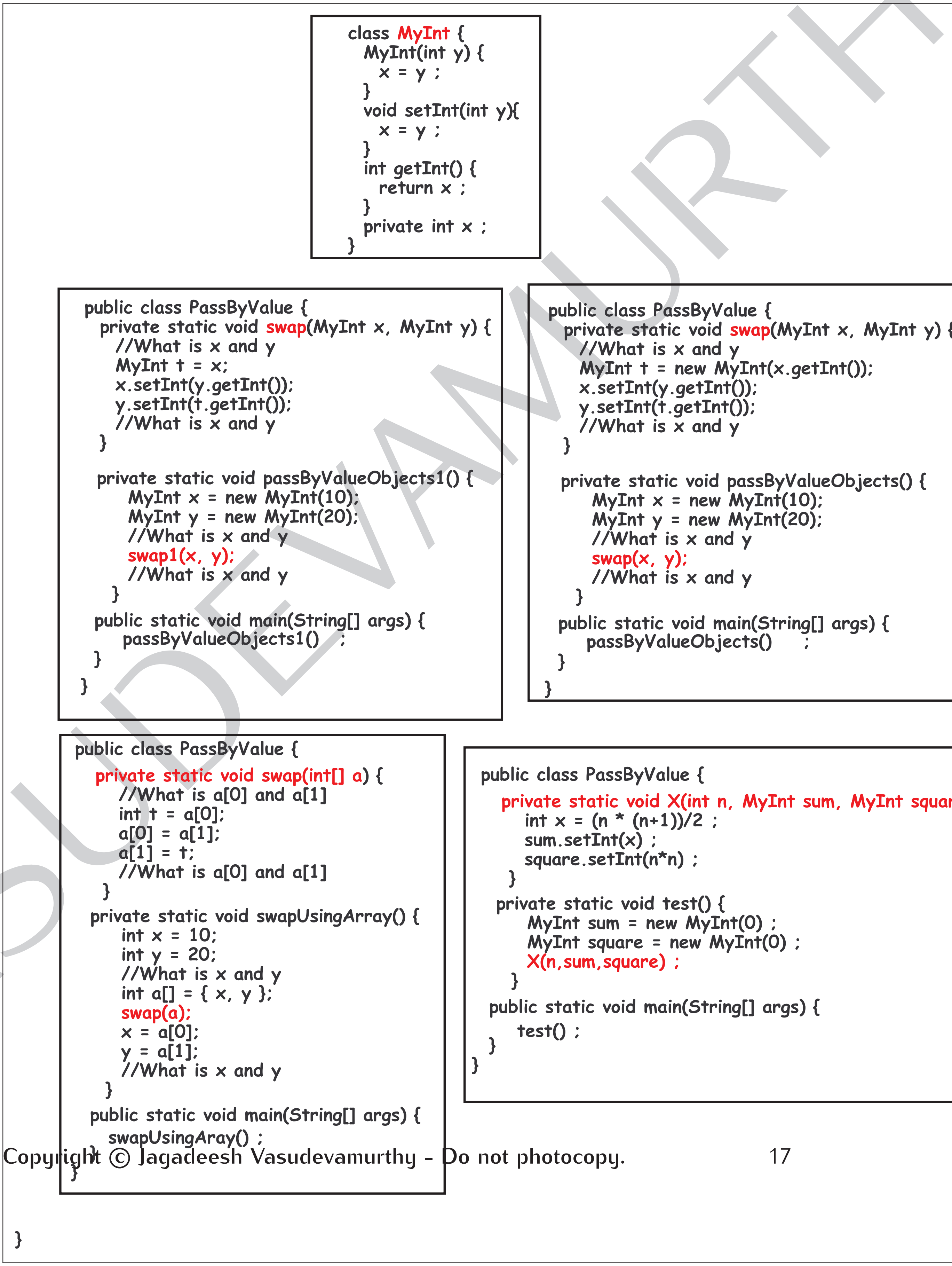
```
public class PassByValue {
  private static void swap1(Integer x, Integer y) {
    //What is x and y
    Integer t = new Integer(x);
    x = y;
    y = t;
    //What is x and y
  }

  private static void passByValueObjectTypes {
      Integer x =new Integer(10) ;
      Integer y =new Integer(20) ;
      //What is x and y
      swap1(x, y);
      //What is x and y
    }
  public static void main(String[] args) {
      passByValueObjectTypes();
  }
}
```

```
public class PassByValue {
  private static void swap2(Integer x, Integer y) {
    //What is x and y
    Integer t = new Integer(x);
    x = new Integer(y);
    y = new Integer(t) ;
    //What is x and y
  }

  private static void passByValueObjectTypesThatDoesNotWork() {
      Integer x =new Integer(10) ;
      Integer y =new Integer(20) ;
      //What is x and y
      swap2(x, y);
      //What is x and y
    }
  public static void main(String[] args) {
      passByValueObjectTypesThatDoesNotWork() ;
  }
}
```

Integer objects are immutable,
so you cannot modify the value once they have been created.
You will need to create a new Integer and replace the existing one.

Figure 1.2: Pass by value - Primitive types

1.2.2 Pass by value - Objects



```
class MyInt {
  MyInt(int y) {
    x = y ;
  }
  void setInt(int y){
    x = y ;
  }
  int getInt() {
    return x ;
  }
  private int x ;
}
```

```
public class PassByValue {
  private static void swap(MyInt x, MyInt y) {
    //What is x and y
    MyInt t = x;
    x.setInt(y.getInt());
    y.setInt(t.getInt());
    //What is x and y
  }

  private static void passByValueObjects1() {
      MyInt x = new MyInt(10);
      MyInt y = new MyInt(20);
      //What is x and y
      swap1(x, y);
      //What is x and y
  }
  public static void main(String[] args) {
      passByValueObjects1()   ;
  }
}
```

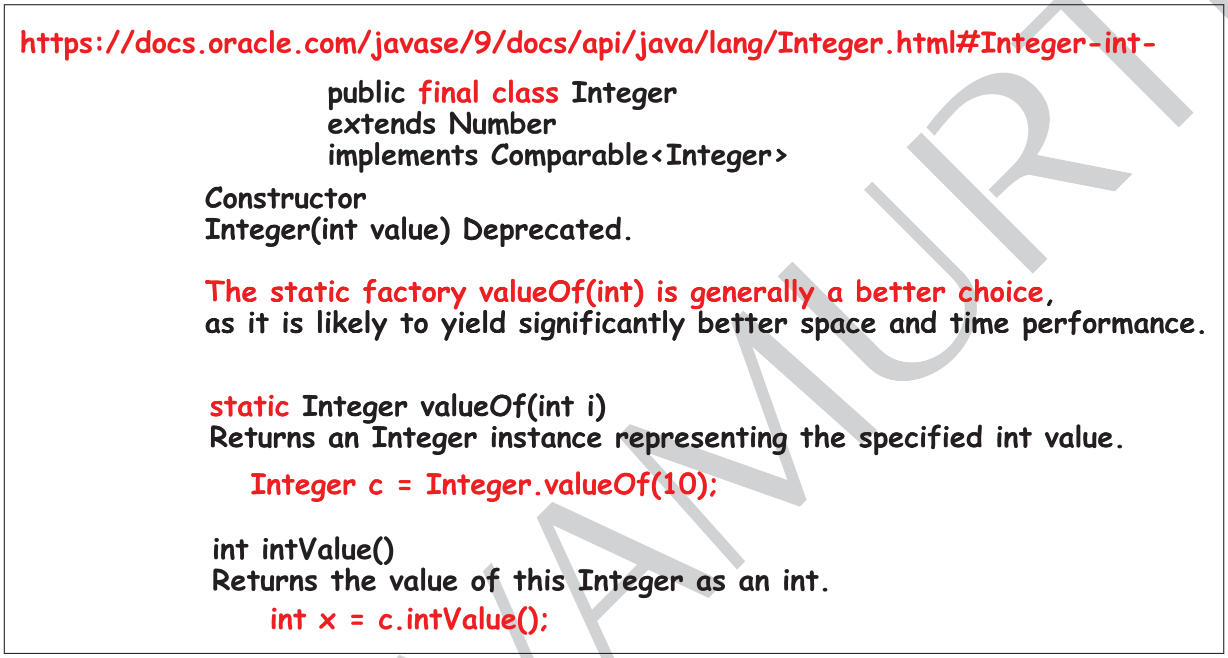
```
public class PassByValue {
  private static void swap(MyInt x, MyInt y) {
    //What is x and y
    MyInt t = new MyInt(x.getInt());
    x.setInt(y.getInt());
    y.setInt(t.getInt());
    //What is x and y
  }

  private static void passByValueObjects() {
      MyInt x = new MyInt(10);
      MyInt y = new MyInt(20);
      //What is x and y
      swap(x, y);
      //What is x and y
  }
  public static void main(String[] args) {
      passByValueObjects()    ;
  }
}
```

```
public class PassByValue {
   private static void swap(int[] a) {
      //What is a[0] and a[1]
      int t = a[0];
      a[0] = a[1];
      a[1] = t;
      //What is a[0] and a[1]
   }
  private static void swapUsingArray() {
      int x = 10;
      int y = 20;
      //What is x and y
      int a[] = { x, y };
      swap(a);
      x = a[0];
      y = a[1];
      //What is x and y
   }
  public static void main(String[] args) {
    swapUsingAray() ;
  }
}
```

```
public class PassByValue {
   private static void X(int n, MyInt sum, MyInt squar
      int x = (n * (n+1))/2 ;
      sum.setInt(x) ;
      square.setInt(n*n) ;
   }
  private static void test() {
      MyInt sum = new MyInt(0) ;
      MyInt square = new MyInt(0) ;
      X(n,sum,square) ;
   }
  public static void main(String[] args) {
     test() ;
  }
}
```

```
}
```

1.2.3 Integer



https://docs.oracle.com/javase/9/docs/api/java/lang/Integer.html#Integer-int-

```
public final class Integer
extends Number
implements Comparable<Integer>
```

Constructor
Integer(int value) Deprecated.

The static factory valueOf(int) is generally a better choice,
as it is likely to yield significantly better space and time performance.

static Integer valueOf(int i)
Returns an Integer instance representing the specified int value.

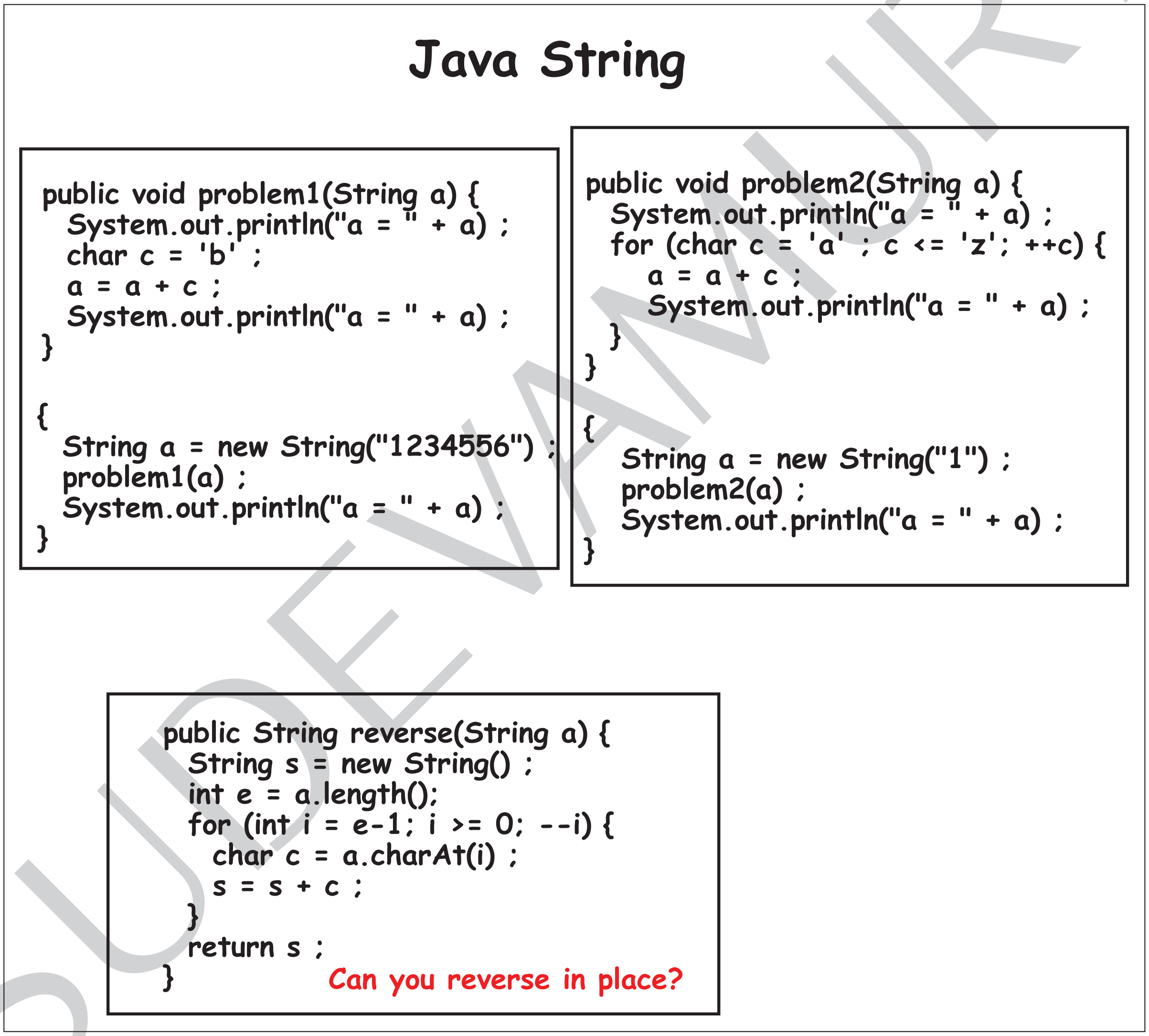
```
Integer c = Integer.valueOf(10);
```

int intValue()
Returns the value of this Integer as an int.

```
int x = c.intValue();
```

Figure 1.4: Integer

1.3 Problems with String



Java String

```
public void problem1(String a) {
  System.out.println("a = " + a) ;
  char c = 'b' ;
  a = a + c ;
  System.out.println("a = " + a) ;
}

{
  String a = new String("1234556") ;
  problem1(a) ;
  System.out.println("a = " + a) ;
}
```

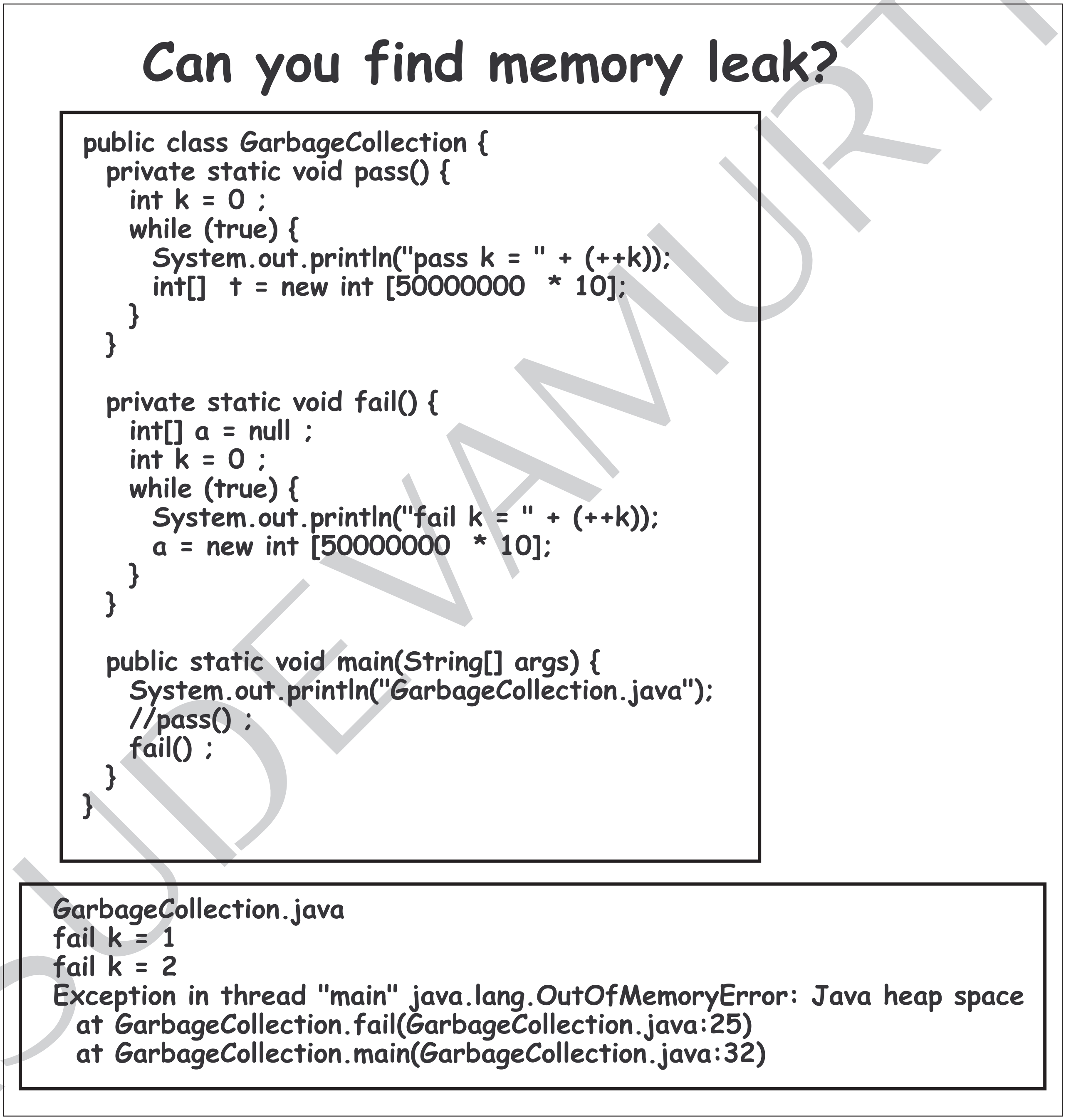
```
public void problem2(String a) {
  System.out.println("a = " + a) ;
  for (char c = 'a' ; c <= 'z'; ++c) {
     a = a + c ;
     System.out.println("a = " + a) ;
  }
}

{
   String a = new String("1") ;
   problem2(a) ;
   System.out.println("a = " + a) ;
}
```

```
public String reverse(String a) {
  String s = new String() ;
  int e = a.length();
  for (int i = e-1; i >= 0; --i) {
    char c = a.charAt(i) ;
    s = s + c ;
  }
  return s ;
}
```

Can you reverse in place?

Figure 1.5: What is the problem ?

1.4 Memory leak in Java



Can you find memory leak?

```
public class GarbageCollection {
  private static void pass() {
    int k = 0 ;
    while (true) {
      System.out.println("pass k = " + (++k));
      int[]  t = new int [50000000  * 10];
    }
  }

  private static void fail() {
    int[] a = null ;
    int k = 0 ;
    while (true) {
      System.out.println("fail k = " + (++k));
      a = new int [50000000  * 10];
    }
  }

  public static void main(String[] args) {
    System.out.println("GarbageCollection.java");
    //pass() ;
    fail() ;
  }
}
```

```
GarbageCollection.java
fail k = 1
fail k = 2
Exception in thread "main" java.lang.OutOfMemoryError: Java heap space
  at GarbageCollection.fail(GarbageCollection.java:25)
  at GarbageCollection.main(GarbageCollection.java:32)
```

Figure 1.6: Where is memory leak?