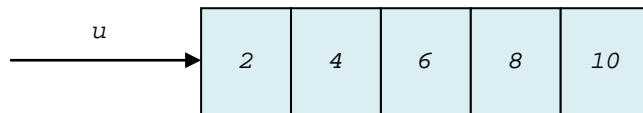


# Exam: solution

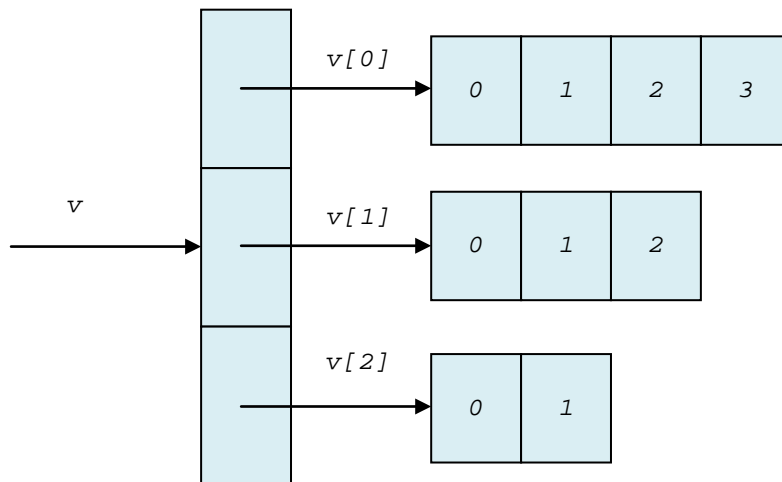
## Tasks: solutions

### Task 1 (2 points + 3 points)

a) (2 points)

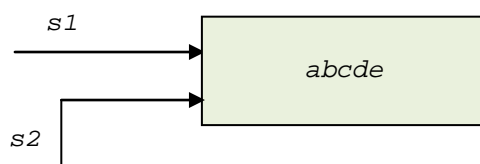


b) (3 points)

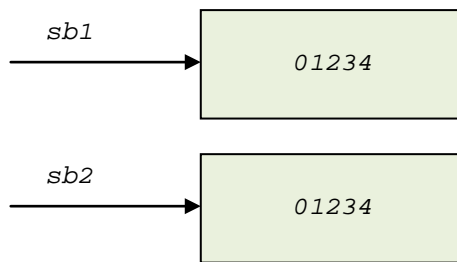


### Task 2 (1 point + 2 points + 2 points)

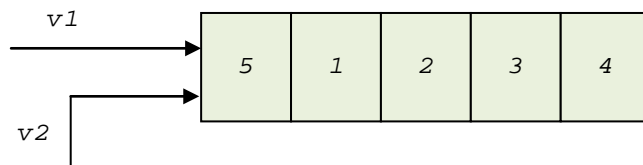
a) (1 point)



b) (2 points)



c) (2 points)



### Task 3 (3 points + 2 points + 2 points)

a) (3 points)

```
public static Point selectPoint (Point p1, Point p2)
{
    Point    p = p1;
    if (p2.distance (Point.ORIGIN) < p1.distance (Point.ORIGIN))
        p = p2;

    return p;
}
```

b) (2 points)

```
public static Point selectPoint (Point p1, Point p2, Point p3, Point p4)
{
    return selectPoint (selectPoint (p1, p2), selectPoint (p3, p4));
}
```

c) (2 points)

```
Point    p1 = new Point (2, 3);
Point    p2 = new Point (1, 4);
Point    p3 = new Point (3, 1);
Point    p4 = new Point (2, 2);

Point    p = selectPoint (p1, p2, p3, p4);
```

### Task 4 (3 points + 3 points + 3 points)

a) (3 points)

```
public String toString ()
{
```

```

    StringBuilder s = new StringBuilder ("{" + this.name + ": [" );
    int gradeIndex = 0;
    for (gradeIndex = 0; gradeIndex < GRADE_COUNT - 1; gradeIndex++)
        s.append (grades[gradeIndex] + ", ");
    s.append (grades[gradeIndex] + "]}");

    return s.toString ();
}

```

b) (3 points)

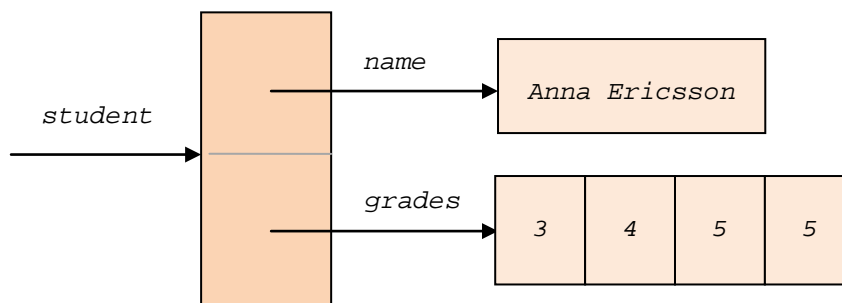
```

public double averageGrade ()
{
    int sum = 0;
    for (int gradeIndex = 0; gradeIndex < GRADE_COUNT; gradeIndex++)
        sum += grades[gradeIndex];
    double average = (double) sum / GRADE_COUNT;

    return average;
}

```

c) (3 points)



## Task 5 (1 point + 2 points + 2 points + 1 point)

a) (1 point)

The classes `LetterSupplierA`, `LetterSupplierB` and `LetterSupplierC` implement the interface `LetterSupplier` (the class `LetterSupplierC` is a subclass to class `LetterSupplierA`, and thus implements the interface – it inherits the method `lowerCaseLetter` from the superclass `LetterSupplierA`). The references in the vector `ls` are of type `LetterSupplier`, and can refer to objects of all implementing classes.

b) (2 points)

A reference of type `LetterSupplier` activates the method in the definition class of the appointed object. This gives the following printout:

```

a | A
b | B
a | C

```

c) (2 points)

The class `LetterSupplierC` is a subclass to `LetterSupplierA`. Therefore, the reference `lsa1` of type `LetterSupplierA` can refer to the created object of type `LetterSupplierC`. The superclass reference `lsa1` activates the subclass' method `upperCaseLetter`, and the following printout is created:

C

If statement (1) is included a compilation error occurs. The method `letter` does not exist in the superclass `LetterSupplierA`, and the reference `lsa1` cannot activate that method. The method can be activated with a reference of type `LetterSupplierC`.

d) (1 point)

The class `LetterSupplierB` is not a subclass to class `LetterSupplierA`. The reference `lsa2` of type `LetterSupplierA` cannot refer to the created object of type `LetterSupplierB`. A compilation error occurs.

## Task 6 (2 points + 2 points + 5 points)

a) (2 points)

To determine the Fibonacci number in position  $n$ , all preceding Fibonacci numbers are determined and stored – there are  $n$  such numbers. The memory complexity of the algorithm is:

$$m(n) = n$$

$$m(n) \in \mathcal{O}(n)$$

b) (2 points)

The time complexity in terms of additions can be expressed with the following complexity function:

$$t(n) = 0, n \leq 1$$

$$t(n) = n - 1, n > 1$$

$$t(n) \in \mathcal{O}(n)$$

c) (5 points)

```
public static long fibonacciNumber (int n)
{
    long    f = 0;
    if (n == 0)
        f = 0;
    else if (n == 1)
        f = 1;
    else
    {
        long    f1 = 0;
        long    f2 = 1;
        for (int pos = 2; pos <= n; pos++)
        {
            f = f1 + f2;
            f1 = f2;
            f2 = f;
        }
    }

    return f;
}
```

The algorithm in this method uses in the worst case two extra memory cells (`f1` and `f2`). The memory complexity of the algorithm is:

$$m(n) = 0, n \leq 1$$

$$m(n) = 2, n > 1$$

$$m(n) \in \Theta(1)$$