Introduction to Java

Coloring- and Workbook

Version: August 2, 2020

Initial concept by Christian Clausen

Contents

Chapter 1

Terminology

Matching 1. Connect the terms on the left to the closest matching code on the right. Each term should be connected to one line of code, and vice versa.

Definition	`A a;`
Declaration	`new A()`
Initialization	`a = new A();`
Instantiation	`A a = new A();`

Coloring 2. Highlight the **keywords** in the follow code:

```
1 public class HelloWorld {
2   public static void main(String[] args) {
3     System.out.println("Hello, World!");
4   }
5 }
```

Elimination 3. Complete this text by crossing out the incorrect option:

We pass **arguments/parameters** to functions when we call them. The **arguments/parameters** declare the types, so when we call a function the **arguments/parameters** should be expressions of those types. When we **ini-**

tialize/instantiate/declare an object the constructor is invoked. Then we can also pass **arguments/parameters** to that constructor.

Matching 4. Connect the terms on the left to the closest matching code on the right. Each term should be connected to one line of code, and vice versa.

Assignment	`new A()`
Expression	`a = new A()`
Statement	`A a = new A();`

Definition 5. Mark all that apply:

	Assignment	Increment	Expression
`a++`			
`a = 1`			
`a + 1`			
`a == 1`			
`a += 1`			
`a = a + 1`			
`a == a + 1`			

Coloring 6. Syntax color the follow code:

☐ Locals ☐ Parameters ☐ Fields

```
1 public class Circle {
2   public static final float PI = 3.14f;
3   private int radius;
4   public Circle(final int r) {
5     if (r == 0)
6       this.radius = 1;
```

```
7       else
8         this.radius = r;
9     }
10    private float area() {
11      float result = radius * radius * PI;
12      return result;
13    }
14    public static void main(String[] args) {
15      try {
16        Circle c = new Circle(args[0]);
17        System.out.println("Area: " + c.area());
18      } catch (ArrayIndexOutOfBoundsException e) {
19        System.out.println("Usage: java Circle [radius]");
20      }
21    }
22  }
```

Matching 7. Connect the terms on the left to the closest matching code on the right. Each term should be connected to one line of code, and vice versa.

Assignment	`a++`
Expression	`a + 1`
Incrementation	`a += 1;`
Statement	`a = a + 1`

Completion 8. Place each of the following labels twice to make the sentences true:

- Statements
- Expressions
- Assignments

All _____ are _____.

All _____ can be _____.

But not all _____ can be _____.

Matching 9. Connect the terms on the left to the closest matching code on the right. Each term should be connected to one line of code, and vice versa.

Incrementation	`a++`
Initialization	`new A()`
Instantiation	`A a = new A()`
Iteration	`for (A a : array);`

Labeling 10. Place the following labels:

- Instance
- Instantiation
- Instance variable

```
0 public class Instance {
1   private A a; ———— [          ]
2   public void foo() {
3     a = new A(); ———— [          ]
4   }
5 }
```

Coloring 11. Syntax color the follow code:

☐ Statement keywords ☐ Other keywords

```
1 public class Circle {
2   public static final float PI = 3.14f;
```

```
  private int radius;
  public Circle(final int r) {
    if (r == 0)
      this.radius = 1;
    else
      this.radius = r;
  }
  private float area() {
    float result = radius * radius * PI;
    return result;
  }
  public static void main(String[] args) {
    try {
      Circle c = new Circle(args[0]);
      System.out.println("Area: " + c.area());
    } catch (ArrayIndexOutOfBoundsException e) {
      System.out.println("Usage: java Circle [radius]");
    }
  }
}
```

Chapter 2

Types

Matching 12. Connect the terms on the left to the closest matching code on the right. Each term should be connected to one line of code, and vice versa.

`boolean`	`"A"`
`byte`	`'A'`
`char`	`-10`
`double`	`20L`
`float`	`true`
`int`	`null`
`long`	`10000`
`Object`	`1.5690`
`short`	`127000`
`String`	`3.1415f`

Coloring 13. Syntax color the follow code:

☐ Primitive types ☐ Other types

```
1 public class Circle {
2   public static final float PI = 3.14f;
```

```
3   private int radius;
4   public Circle(final int r) {
5     if (r == 0)
6       this.radius = 1;
7     else
8       this.radius = r;
9   }
10  private float area() {
11    float result = radius * radius * PI;
12    return result;
13  }
14  public static void main(String[] args) {
15    try {
16      Circle c = new Circle(args[0]);
17      System.out.println("Area: " + c.area());
18    } catch (ArrayIndexOutOfBoundsException e) {
19      System.out.println("Usage: java Circle [radius]");
20    }
21  }
22 }
```

Definition 14. Mark as precisely as possible which type of variable each expression can be assigned to:

	int	int[]	Integer	Object
5				
5L				
5.0				
'5'				
"5"				
null				
new int[1]				

Chapter 3

Accessibility

Definition 15. Mark as precisely as possible the accessibility for each of the four visibility keywords:

	`public`	`private`	`protected`	`package private`
Same object				
Same class				
Same file				
Inner classes				
Outer classes				
Subclasses				
Superclasses				
Same package				
Everywhere				

Chapter 4

UML

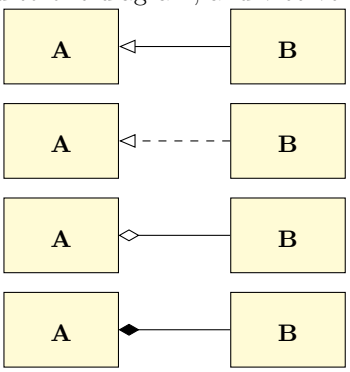
Matching 16. Connect the terms on the left to the closest matching diagram on the right. Each term should be connected to one diagram, and vice versa.

Aggregation

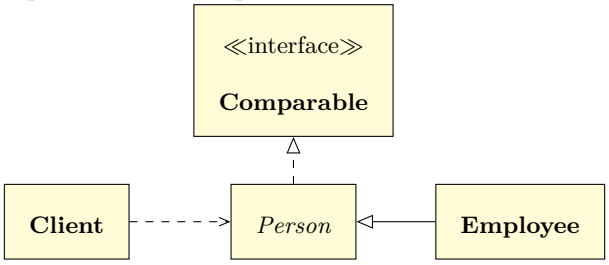
Composition

Implementation

Inheritance



Completion 17. Complete the code, based on the UML class diagram.




```
0 abstract class Person [          ] Comparable<Person> {
1 }
```

```
2 class Employee [          ] Person {
3 }
4 public class Client {
5   private [          ] owner;
6   public Client(){
7     owner = new [          ]();
8   }
9 }
```

Chapter 5

Loops

Elimination 18. Cross out everything that is not a type of loop in Java:

while	for	until
do-while	do-for	do-until
foreach	for-in	if

Completion 19. Complete this for-loop to make the code work:

```
0 int[] arr = new int[] { 1, 2, 3, 4, 5 };
1 int sum = 0;
2 for (int i = [          ]; [          ]; [          ]) {
3   sum += arr[i];
4 }
5 System.out.println(sum);
```

Completion 20. Complete this for-loop to make the code print the numbers in *reverse*:

```
0 int[] arr = new int[] { 1, 2, 3, 4, 5 };
1 String result = "";
2 for (int i = [          ]; [          ]; [          ]) {
```

```
3   if (result.length() != 0) result += " ";
4   result += arr[i];
5 }
6 System.out.println(result);
```