DEV4 - Exam sample 1

The INFDEV team

1 Question 1

Given the following class definitions, and a piece of code that uses them, fill in the stack, heap, and PC with all steps taken by the program at runtime.

- Points: *3 (30% of total)*.
- Grading: one point per correctly filled-in execution step.
- Associated learning objective: *abstraction*.

```
1  interface NumberVisitor {
2    void OnInt();
3    void OnFloat();
4  }
5  interface Number {
6    void Visit(NumberVisitor visitor);
7  }
8  class TypePrettyPrinter : NumberVisitor {
9    TypePrettyPrinter() {
10   }
11   void OnInt() {
12     Console.WriteLine("I␣am␣Int");
13   }
14   void OnFloat() {
15     Console.WriteLine("I␣am␣Float");
16   }
17 }
18 class MyFloat : Number {
19   MyFloat() {
20   }
21   void Visit(NumberVisitor visitor) {
22     visitor.OnFloat();
23   }
24 }
25 class MyInt : Number {
26   MyInt() {
27   }
28   void Visit(NumberVisitor visitor) {
29     visitor.OnInt();
30   }
31 }
32 TypePrettyPrinter visitor = new TypePrettyPrinter();
33 Number aNumber = new MyFloat();
34 aNumber.Visit(visitor);
35 ...
```

1. Stack:

PC
1

2. Stack:

PC
32

   Heap:

1
__type=TypePrettyPrinter

3. Stack:

PC	...		PC	ret	this
32	...		10	null	ref 1

   Heap:

1
__type=TypePrettyPrinter

4. Stack:

PC	...		PC	ret
32	...		10	ref 1

Heap:

1
__type=TypePrettyPrinter

5. Stack:

PC	visitor
33	ref 1

Heap:

1
__type=TypePrettyPrinter

6. Stack:

PC	visitor
33	ref 1

Heap:

1	2
__type=TypePrettyPrinter	__type=MyFloat

7. Stack:

PC	...		PC	ret	this
33	...		20	null	ref 2

Heap:

1	2
__type=TypePrettyPrinter	__type=MyFloat

8. Stack:

PC	...		PC	ret
33	...		20	ref 2

Heap:

1	2
__type=TypePrettyPrinter	__type=MyFloat

9. Stack:

PC	aNumber	visitor
34	ref 2	ref 1

Heap:

1	2
__type=TypePrettyPrinter	__type=MyFloat

10. Stack:

PC	...		PC	ret	this	visitor
34	...		22	null	ref 2	ref 1

Heap:

1	2
__type=TypePrettyPrinter	__type=MyFloat

11. Stack:

PC	...		PC	...		PC	ret	this
34	...		22	...		15	null	ref 1

Heap:

1	2
__type=TypePrettyPrinter	__type=MyFloat

12. Stack:

PC	...		PC	...		PC	ret
34	...		22	...		15	null

Heap:

1	2
__type=TypePrettyPrinter	__type=MyFloat

Output: `"I am Float"`

13. Stack:

PC	...		PC	ret
34	...		22	null

Heap:

1	2
__type=TypePrettyPrinter	__type=MyFloat

Output: `"I am Float"`

14. Stack:

PC	aNumber	visitor
35	ref 2	ref 1

Heap:

1	2
__type=TypePrettyPrinter	__type=MyFloat

Output: `"I am Float"`

2 Question 2

Given the following class definitions, and a piece of code that uses them, fill in the declarations, class definitions, and PC with all steps taken by the compiler while type checking.

```
1  interface NumberVisitor {
2    void OnInt();
3    void OnFloat();
4  }
5  interface Number {
6    void Visit(NumberVisitor visitor);
7  }
8  class TypePrettyPrinter : NumberVisitor {
9    TypePrettyPrinter() {
10   }
11   void OnInt() {
12     Console.WriteLine("I␣am␣Int");
13   }
14   void OnFloat() {
15     Console.WriteLine("I␣am␣Float");
16   }
17 }
18 class MyFloat : Number {
19   MyFloat() {
20   }
21   void Visit(NumberVisitor visitor) {
22     visitor.OnFloat();
23   }
24 }
25 class MyInt : Number {
26   MyInt() {
27   }
28   void Visit(NumberVisitor visitor) {
29     visitor.OnInt();
30   }
31 }
32 NumberVisitor visitor = new TypePrettyPrinter();
33 Number aNumber = new MyFloat();
34 aNumber.Visit(visitor);
35 ...
```

1. Declarations:

PC
1

2. Declarations:

PC
5

Classes:

NumberVisitor
OnFloat=NumberVisitor → void OnInt=NumberVisitor → void

3. Declarations:

PC
8

Classes:

Number	NumberVisitor
Visit=(Number×NumberVisitor) → void	OnFloat=NumberVisitor → void OnInt=NumberVisitor → void

4. Declarations:

PC
17

Classes:

Number	NumberVisitor	TypePrettyPrinter
Visit=(Number×NumberVisitor) → void	OnFloat=NumberVisitor → void OnInt=NumberVisitor → void	OnFloat=TypePrettyPrinter → void OnInt=TypePrettyPrinter → void TypePrettyPrinter=TypePrettyPrinter → TypePrettyPrinter

5. Declarations:

PC
24

Classes:

MyFloat	Number	NumberVisitor	TypePrettyPrinter
MyFloat=MyFloat → MyFloat Visit=(MyFloat×NumberVisitor) → void	Visit=(Number×NumberVisitor) → void	OnFloat=NumberVisitor → void OnInt=NumberVisitor → void	OnFloat=TypePrettyPrinter → void OnInt=TypePrettyPrinter → void TypePrettyPrinter=TypePrettyPrinter → TypePrettyPrinter

6. Declarations:

PC
31

Classes:

MyFloat	MyInt	Number	NumberVisitor	TypePrettyPrinter
MyFloat=MyFloat → MyFloat Visit=(MyFloat×NumberVisitor) → void	MyInt=MyInt → MyInt Visit=(MyInt×NumberVisitor) → void	Visit=(Number×NumberVisitor) → void	OnFloat=NumberVisitor → void OnInt=NumberVisitor → void	OnFloat=TypePrettyPrinter → void OnInt=TypePrettyPrinter → void TypePrettyPrinter=TypePrettyPrinter → TypePrettyPrinter

7. Declarations:

PC	visitor
33	NumberVisitor

Classes:

MyFloat	MyInt	Number	NumberVisitor	TypePrettyPrinter
MyFloat=MyFloat → MyFloat Visit=(MyFloat×NumberVisitor) → void	MyInt=MyInt → MyInt Visit=(MyInt×NumberVisitor) → void	Visit=(Number×NumberVisitor) → void	OnFloat=NumberVisitor → void OnInt=NumberVisitor → void	OnFloat=TypePrettyPrinter → void OnInt=TypePrettyPrinter → void TypePrettyPrinter=TypePrettyPrinter → TypePrettyPrinter

8. Declarations:

PC	aNumber	visitor
34	Number	NumberVisitor

Classes:

MyFloat	MyInt	Number	NumberVisitor	TypePrettyPrinter
MyFloat=MyFloat → MyFloat Visit=(MyFloat×NumberVisitor) → void	MyInt=MyInt → MyInt Visit=(MyInt×NumberVisitor) → void	Visit=(Number×NumberVisitor) → void	OnFloat=NumberVisitor → void OnInt=NumberVisitor → void	OnFloat=TypePrettyPrinter → void OnInt=TypePrettyPrinter → void TypePrettyPrinter=TypePrettyPrinter → TypePrettyPrinter

9. Declarations:

aNumber	visitor		PC	ret	arg_1	this
Number	NumberVisitor		34	null	NumberVisitor	Number

Classes:

MyFloat	MyInt	Number	NumberVisitor	TypePrettyPrinter
MyFloat=MyFloat → MyFloat Visit=(MyFloat×NumberVisitor) → void	MyInt=MyInt → MyInt Visit=(MyInt×NumberVisitor) → void	Visit=(Number×NumberVisitor) → void	OnFloat=NumberVisitor → void OnInt=NumberVisitor → void	OnFloat=TypePrettyPrinter → void OnInt=TypePrettyPrinter → void TypePrettyPrinter=TypePrettyPrinter → TypePrettyPrinter

10. Declarations:

aNumber	visitor		PC	ret	arg_1	this
Number	NumberVisitor		34	void	NumberVisitor	Number

Classes:

MyFloat	MyInt	Number	NumberVisitor	TypePrettyPrinter
MyFloat=MyFloat → MyFloat Visit=(MyFloat×NumberVisitor) → void	MyInt=MyInt → MyInt Visit=(MyInt×NumberVisitor) → void	Visit=(Number×NumberVisitor) → void	OnFloat=NumberVisitor → void OnInt=NumberVisitor → void	OnFloat=TypePrettyPrinter → void OnInt=TypePrettyPrinter → void TypePrettyPrinter=TypePrettyPrinter → TypePrettyPrinter

11. Declarations:

PC	aNumber	visitor
35	Number	NumberVisitor

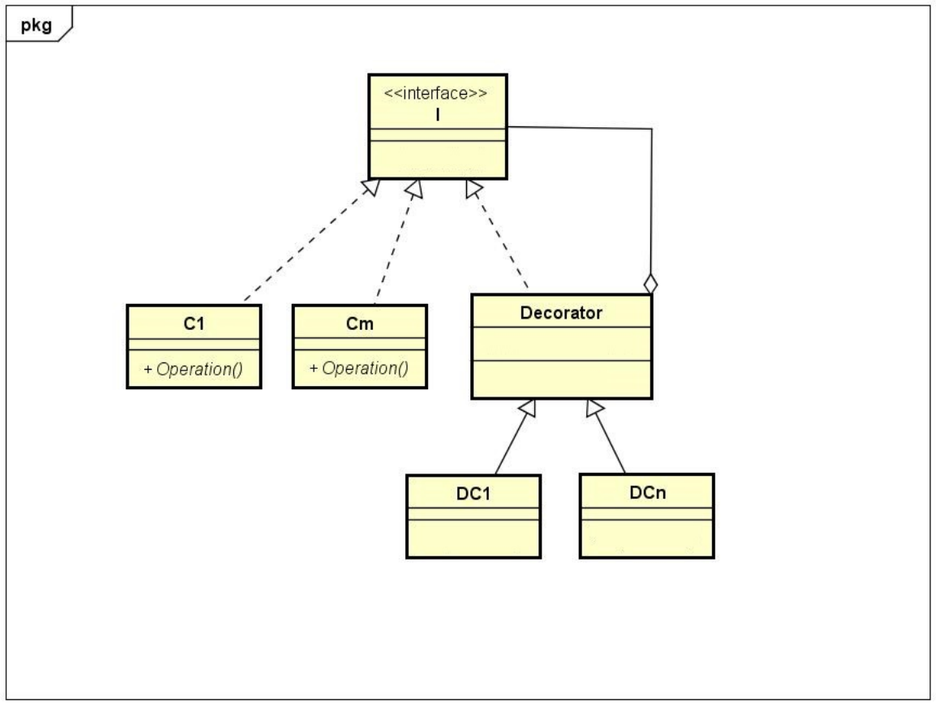
Classes:

MyFloat	MyInt	Number	NumberVisitor	TypePrettyPrinter
MyFloat=MyFloat → MyFloat Visit=(MyFloat×NumberVisitor) → void	MyInt=MyInt → MyInt Visit=(MyInt×NumberVisitor) → void	Visit=(Number×NumberVisitor) → void	OnFloat=NumberVisitor → void OnInt=NumberVisitor → void	OnFloat=TypePrettyPrinter → void OnInt=TypePrettyPrinter → void TypePrettyPrinter=TypePrettyPrinter → TypePrettyPrinter

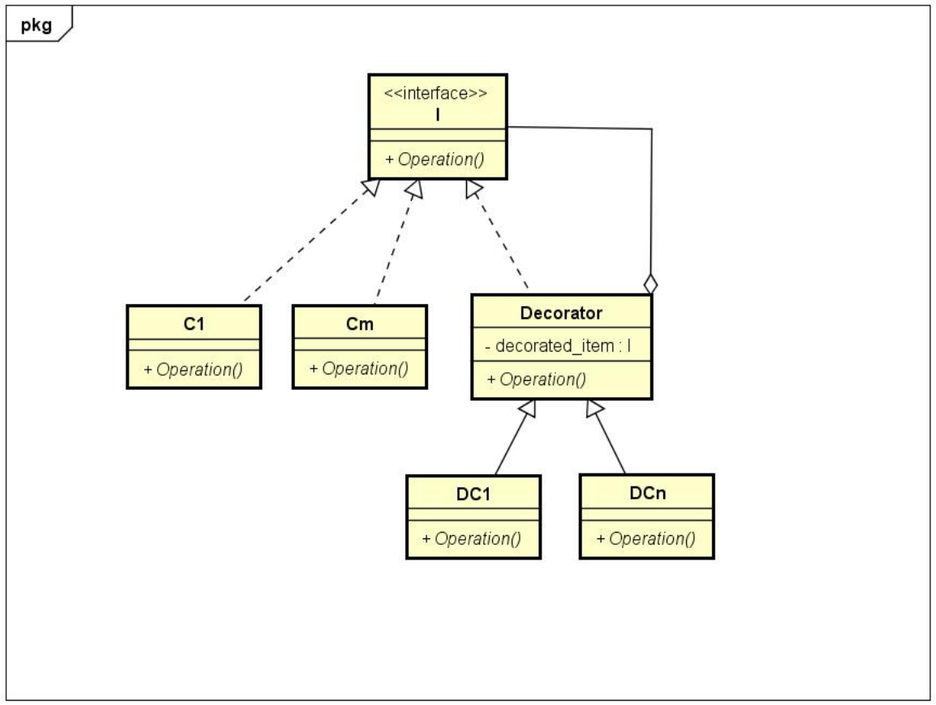
- Points: *3 (30% of total)*.
- Grading: one point per correctly filled-in type checking step.
- Associated learning objective: *type checking*.

3 Question 3

Given the following UML diagram of the decorator method, fill in the missing parts.



For the solution see below:



- Points: *4 (40% of total)*.
- Grading: one point per correctly filled-in part.
- Associated learning objective: *abstraction*.