

CS2 Intense Study Session

PART 1: INNER CLASSES AND INTERFACES

Exercise 1

1. The following code has one mistake. Can you spot it? What will be the output of the code once the error is corrected (do not change the classes body)?
2. Edit class X so that it will output the name defined in the test class

```
class test{
    private String name;
    public test(String name){
        this.name = name;
    }
    public static void main(String[] args){
        test t = new test("Steven");
        test.X t2 = t.new X("Pietro");
        t2.yo();
        test.Y.notYo();
        test.Y t3 = new test.Y("Jan");
        t3.notYo();
    }
    class X{
        private String name;
        public X(String name){
            this.name = name;
        }
        public void yo(){
            System.out.println("Yo "+name);
        }
    }
    static class Y{
        private String name;
        public Y(String name){
            this.name = name;
        }
        public void notYo(){
            System.out.println("Not yo "+name);
        }
    }
}
```

Answers exercise 1:**1.**

The mistake is in the line `test.Y.notYo()`; the method `notYo()` is not static therefore cannot be instantiated. That line should simply be removed or commented.

The code will output:

Yo Pietro

Not yo Jan

2.

To print the name of the parent's class, simply edit this

```
class X{  
    private String name = test.this.name;  
    ...  
}
```

The output will then be:

Yo Steven

Not yo Jan

Exercise 2

Look at the code below.

1. Will this code compile? Will it run?
2. How many classes are there in the code? What are their parent classes?
3. What happens if add the line `x.hey();`?
4. Will the code work if you remove the class Hello?
5. Edit the inner class to print “hey” instead of “hello” without changing the first Hello class

```
class Hello{
    public void hello(){
        System.out.println("Hello");
    }
}
class test{
    public static void main(String[] args){
        Hello x = new Hello(){
            public void hey(){
                System.out.println("Hey");
            }
        };
        x.hello();
    }
}
```

Answers exercise 2:

1. The code will compile and output "Hello"
2. There are 3 classes: "test", "Hello" and a [Local Class](#) "Hello". The Local Class Hello is a child of Hello
3. There will be a compiler error as x is an instance of Hello and Hello doesn't have the method hey()
4. No, as Hello isn't defined
- 5.

```
    Hello x = new Hello(){  
        public void hello(){  
            System.out.println("Hey");  
        }  
    };
```

Exercise 2b

!!! DISCLAIMER: Your brain will now hurt a lot. !!!

Look at the code below

- 1. Will it compile? If no, explain why. If yes, what will be the output?**

```
interface X{
    public void hello();
}
class Hello implements X{
    public void hello(){
        System.out.println("Hello");
    }
}
class test{
    public static void main(String[] args){
        X x = new Hello(){
            public void hey(){
                System.out.println("Hey");
            }
        };
        x.hello();
    }
}
```

Answers exercise 2b:

1. Yes, it will compile, and output "Hello". The reason is that the "new Hello()" class extends the Hello class, which implements X. Just look at the exercise before and you'll hopefully understand a bit more.

Exercise 3

Last exercise on nested classes.

Look at the code below.

1. What will be the output of the code?
2. Change the code, if possible, so that it will output the message of the other class B

```
class B{
    public void seeTheFuture(){
        System.out.println("You will fail CS2");
    }
}
class A{
    public static void main(String[] args){
        B future = new B();
        future.seeTheFuture();
    }
    static class B{
        public void seeTheFuture(){
            System.out.println("You will pass CS2");
        }
    }
}
```


Answers exercise 3:

1. The output of the code is..... "You will pass CS2"!! (don't get too excited)
2. It is impossible to print "You will fail CS2" unless you change the name of the inner class B. So chill down, "You will pass CS2" (jk keep studying)

Exercise 4

1. **(True/False)** An abstract class cannot contain a `main(String[] args)` method.
2. **(True/False)** Only inner and local classes can be static
3. **(True/False)** An abstract method cannot be overloaded
4. **(True/False)** Abstract methods can have a body
5. **(True/False)** The following code compiles successfully:

```
abstract class A{
    abstract protected void george();
}
class B extends A{
    public void george(){
        System.out.println("Hello George");
    }
}
```

6. **(True/False)** Variables can also be abstract
7. **(True/False)** abstract methods cannot be final
8. **(True/False)** Interfaces can contain variables

Answers exercise 4:

1. False. In fact, interfaces can also have a main method.
2. True,
3. True,
4. False,
5. True
6. False
7. True
8. True, and the variables will be final.

Exercise 5

Write down 3 differences between interfaces and abstract classes.

Look at the code below.

1. What will be the output of this code?

```
abstract class A{
    abstract protected void george();
    int i = 0;
    public static void main(String[] args){
        A b = new B();
        A l = new L();
        b.george();
        b.george();
        l.george();
    }
}
class B extends A{
    public void george(){
        i += 1;
        System.out.println("Hello George "+i);
    }
}
class L extends A{
    public void george(){
        System.out.println("Hello George "+i);
    }
}
```

2. How would you change class A to obtain this:

Hello George 1

Hello George 2

Hello George 2

Answers exercise 5

Abstract classes can have protected methods, you can implement multiple interfaces but only one abstract class, abstract classes can have non-final variables...

1. The output will be:

Hello George 1


Hello George 2


Hello George 0


2. Simply set i to static

Exercise 6

The compiler generates these 3 .class files. Simply describe what this means

 ASB.class

 A.class

 C.class

Answer exercise 6: There are 3 classes: A, B and C. B is an inner class of A.