

Exercise 6

Inheritance

November 5, 2021

Task 1

Consider the following Java code:

```
class A {
    String get(Client a) { return "AC"; }
}

class B extends A {
    String get(SpecialClient a) { return "BS"; }
}

class C extends B {
    String get(Client a) { return "CC"; }
    String get(SpecialClient a) { return "CS"; }
}

class Client {
    String m(A x, A y) { return "C1" + x.get(this) + y.get(this); }
    String m(C x, A y) { return "C2" + x.get(this) + y.get(this); }
    String m(B x, A y) { return "C3" + x.get(this) + y.get(this); }
    String m(C x, C y) { return "C4" + x.get(this) + y.get(this); }
}

class SpecialClient extends Client {
    String m(A x, A y) { return "S1" + x.get(this) + y.get(this); }
    String m(C x, A y) { return "S2" + x.get(this) + y.get(this); }
    String m(B x, A y) { return "S3" + x.get(this) + y.get(this); }
    String m(B x, C y) { return "S4" + x.get(this) + y.get(this); }
}

public class Main {
    public static void main(String[] args) {
        Client client = new SpecialClient();
        C c = new C();
        B b = c;
        System.out.println(client.m(b, c));
    }
}
```

What is the result of compiling the code and running the `Main.main` method?

- (a) The program does not compile due to a type error
- (b) The program prints a string starting with "S4"
- (c) The program prints a string ending with "CS"

- (d) The program prints a string containing “BS”
- (e) None of the above

Task 2 Overloading and Overriding

Consider the following class in Java:

```
public class Person {  
  
    protected double salary;  
  
    public Person(double salary) {  
        this.salary = salary;  
    }  
  
    public boolean haveSameIncome(Person other) {  
        return this.salary == other.getIncome();  
    }  
  
    public double getIncome() {  
        return salary;  
    }  
}
```

Consider also the following subclass of Person, a person with a spouse, which takes the salary of the spouse into account as well:

```
public class MarriedPerson extends Person {  
  
    private double spouseSalary;  
  
    public MarriedPerson(double salary, double spouseSalary) {  
        super(salary);  
        this.spouseSalary = spouseSalary;  
    }  
  
    public boolean haveSameIncome(MarriedPerson other) {  
        return this.getIncome() == other.getIncome();  
    }  
  
    public double getIncome() {  
        return ((salary + spouseSalary) / 2);  
    }  
}
```

A) Show an example with the variables p1 and p2, such that p1.haveSameIncome(p2) returns false, but p1.getIncome() == p2.getIncome() returns true. In other words, fill in the following blank with valid code, such that the assertion below is also valid. Do not use reflection and assume that Person has no other subclasses.

```
Person p1;  
MarriedPerson p2;
```

```
assert (!p1.haveSameIncome(p2) && p1.getIncome() == p2.getIncome());
```

B) Propose changes to `Person` and `MarriedPerson` such that the assertion will fail.

B.1 Can you change **only** `MarriedPerson.haveSameIncome`, such that the assertion will fail for your solution to subtask **A**? If yes, provide the modified method. Otherwise, explain why this is not possible.

B.2 Can you change **only** `Person.haveSameIncome`, such that the assertion will fail for your solution to subtask **A**? If yes, provide the modified method. Otherwise, explain why this is not possible.

Task 3

Consider the following C# classes:

```
public class Matrix {
    public virtual Matrix add(Matrix other) {
        Console.WriteLine("Matrix/Matrix");
        return null;
    }
}

public class SparseMatrix : Matrix {
    public virtual SparseMatrix add(SparseMatrix other) {
        Console.WriteLine("SparseMatrix/SparseMatrix");
        return null;
    }
}

public class MainClass {
    public static void Main(string[] args) {
        Matrix m = new Matrix();
        Matrix s = new SparseMatrix();
        add(m, m);
        add(m, s);
        add(s, m);
        add(s, s);
    }

    public static Matrix add(Matrix m1, Matrix m2) {
        return m1.add(m2);
    }
}
```

A) What is the output of this program? Please explain.

B) Without breaking modularity, change only the body of `MainClass.add` to make it possible to always call the most specific add method from the matrix hierarchy.

Task 4 (from a previous exam)

Consider the following C# code, which compiles and executes without raising exceptions:

```
1 class Ingredient {
2     public void mix(Ingredient i1, Ingredient i2) {
3         Console.WriteLine("Ingredient.mix");
4     }
5 }
6
7 class Milk: Ingredient {
8     public void mix(Egg e, Flour f) {
```

```

9      Console.WriteLine("Milk.mix");
10  }
11 }
12
13 class PowderedMilk: Milk {
14     public void mix(Ingredient i, Flour f) {
15         Console.WriteLine("PowderedMilk.mix");
16     }
17 }
18
19 class Egg: Ingredient {}
20
21 class Flour: Ingredient {}
22
23 class Program {
24     static void mix(Ingredient i1, Ingredient i2, Ingredient i3) {
25         (i1 as dynamic).mix(i2 as dynamic, i3 as dynamic);
26     }
27
28     static void Main() {
29         Ingredient i1 = new PowderedMilk();
30         Ingredient i2 = new Egg();
31         Ingredient i3 = new Flour();
32         mix(i1, i2, i3);
33     }
34 }

```

A) Which is the output of the execution of the method `Program.Main()`?

B) List **all** the casts (from line 25) and **all** the methods that can be removed from the given code, such that it still compiles and when executed produces the output from Task A.

Task 5

Some research languages have symmetric multiple dispatch – methods are defined outside classes, and dispatched dynamically on all arguments regardless of order (no overloading at all). There is no designated receiver for a method but rather all arguments are of the same priority – this is intended to handle binary methods better which are often naturally symmetric. Out of all methods that are statically in scope for a given invocation, the runtime selects the most specific method to dispatch according to all arguments, and so there must be a single best implementation for each possible invocation of a method. The return type is not considered in the implementation selection. When compiling a package the compiler analyzes all types used in the package and all methods and makes sure that for each method and argument types combination there is a single best method to be called; if that is not the case it issues an error. Assume the following three classes in such a language:

```

package integer
class Integer { ... }
Integer add(Integer x, Integer y) { ... }

package natural
import integer.Integer
class Natural extends Integer { ... }
Integer add(Natural x, Integer y) { ... }
Integer add(Integer x, Natural y) { ... }
Natural add(Natural x, Natural y) { ... }

package even
import integer.Integer
class Even extends Integer { ... }

```

```
Integer add(Even    x, Integer y) {...}
Integer add(Integer x, Even    y) {...}
Even    add(Even    x, Even    y) {...}
```

The ellipsis in each class body represents (possibly) private data but no other methods.

Each package compiles successfully on its own.

A user has now written the following client:

```
package client
import even.*
import natural.*

void f(Integer x, Integer y) {
    Integer z = add(x, y);
}
```

- What would be the problem in allowing this client to compile in a type safe multiple dispatch language? Show code that would expose the problem.
- Which requirement could we relax so that this call is valid?
- What could we do in the client package in order to resolve the problem, without modifying other packages and without relaxing the requirement mentioned above?

Task 6

(from a previous exam)

Consider the following C++ program:

```
class X {
public:
    X(int p) : fx(p) {}
    int fx;
};
class Y {
public:
    Y(int p) : fy(p) {}
    int fy;
};
class B : public virtual X, public Y {
public:
    B(int p) : X(p-1), Y(p-2) {}
};
class C : public virtual X, public Y {
public:
    C(int p) : X(p+1), Y(p+1) {}
};
class D : public B, public C {
public:
    D(int p) : X(p-1), B(p-2), C(p+1) {}
};

int main() {
    D* d = new D(5);
    B* b = d;
    C* c = d;
    std::cout << b->fx << b->fy
               << c->fx << c->fy;
    return 0;
}
```

What is the output of running the program?

- (a) 5555
- (b) 2177
- (c) 4147
- (d) 7177
- (e) 7777
- (f) None of the above

Task 7 (from a previous exam)

Consider the following C++ code (recall that default constructors, i.e., constructors without arguments, do not need to be called explicitly in C++):

```
class A {
    public:
        A(int i) { std::cout << "A" << i; }
        A() { std::cout << "A1"; }
        virtual int get() { ... }
};

class B: MODIFIER A {
    public:
        B(int i) : A(i) { std::cout << "B" << i; }
};

class C: MODIFIER A {
    public:
        C(int i) : A(i) { std::cout << "C" << i; }
};

class D: public B, public C {
    public:
        D(int i) : B(i + 10), C(i + 20) { std::cout << "D" << i; }
};
```

Now assume that MODIFIER is replaced by public.

A) Why does the following client code not compile?

```
void client() {
    D* d = new D(5);
    std::cout << d->get();
}
```

B) Add a method to one of the classes so that client compiles.

C) What is the output resulting from the call new D(5) in method client?

D) Now, assume that MODIFIER is replaced by public virtual.

What is the new output resulting from the call new D(5) in method client?