Technical assessment

Q1 - Why would you create an abstract class, if it can have no real instances? One answer possible

- ☐ To have common behavior in derived classes
- ☐ To explore a hypothetical class
- ☐ To prevent unwanted method implementation
- ☐ To reserve memory for an unspecified class type

Q2 - What is the best reason to use a design pattern? One answer possible

- ☐ It will result in code that is more extensible and maintainable
- ☐ It will result in a more compact product.
- ☐ It will speed initial development.
- ☐ It will allow you to add that design pattern to your resume.

Q3 - What is encapsulation? One answer possible

- ☐ Defining classes by focusing on what is important for a purpose
- ☐ Hiding the data and implementation details within a class
- ☐ Making all methods private
- ☐ Using words to define classes

Q4 - Which code creates a new object from the Employee class? Multiple answers possible

- ☐ Employee currentEmployee = Employee.Create();
- ☐ Employee currentEmployee = new Employee();
- ☐ Employee currentEmployee;
- ☐ Employee currentEmployee = Employee.New();

Q5 - When is a constructor executed? One answer possible

- ☐ When an object is created from a class using the new keyword
- ☐ When an class is defined using the class keyword
- ☐ Every time an object is referenced
- ☐ When an object is created from a class using the create keyword

Q6 - Which statement best describes the method of inheritance in OOP? One answer possible

- ☐ Inheritance describes the ability to create new classes based on an existing class.
- ☐ Inheritance means that a group of related properties, methods, and other members are treated as a single unit or object.
- ☐ Inheritance forces a class to have a single responsibility from only one parent.
- ☐ Inheritance means that you will never have multiple classes that can be used interchangeably, even though each class implements the same properties or methods in different ways.

Q7 - Which of the following is NOT an advantage of using getters and setters? One answer possible

- ☐ Getters and setters can speed up compilation.
- ☐ Getters and setters provide encapsulation of behavior.
- ☐ Getters and setters provide a debugging point for when a property changes at runtime.
- ☐ Getters and setters permit different access levels.

Q8 - What does the code shown below demonstrate, and why? One answer possible

```
static void Multiply(int num1, int num2) {};
static void Multiply(double num1, double num2, double num3) {};
static void Multiply(float num1, float num2) {};
```

- ☐ Polymorphism, because each method can perform different task
- ☐ Method overriding, because it display the same method name, different or same parameters, and same return type
- ☐ Method overloading, because it allows the creation of several methods with the same name, wich differ by the type of input via parameter
- ☐ Method overriding, because it display the same method name, different parameters, and same return type

Q9 - What are the dangers of using to much abstraction? One answer possible

- ☐ It can increase code vulnerability
- ☐ It can make code unsafe
- ☐ It can limit code readability
- ☐ It can be safer for coding

Q10 - Why is inheritance used when creating a new class? One answer possible

- ☐ To conserve memory
- ☐ To protect attributes from unwanted changes
- ☐ To separate class behaviour from the more general to the more specific
- ☐ To delegate coding responsibility more efficiently

Q11 - Why is unit testing harder in OOP than functional programming? One answer possible

- ☐ Objects may maintain internal state, which is not easily accessible by the tests.
- ☐ The quality of unit testing frameworks for functional languages is better.
- ☐ OOP promotes code reuse, which means that your tests have to consider more use cases.
- ☐ Object-oriented languages tend to rely on frameworks such as Spring or Hibernate, which make them difficult to test.

Q12 - Which concept best describes the following code? One answer possible

```
public class Rectangle
{
        public double GetArea(double length, double width)
        {
                return length * width;
        }
}

public class Square : Rectangle
{
        public double GetArea(double size)
        {
                return size * size;
        }
}
```

- ☐ Encapsulation
- ☐ Abstraction
- ☐ Overloading
- ☐ Delegation

Q13 - What are the main features of OOP? One answer possible

- ☐ Polymorphism
- ☐ Encapsulation
- ☐ Inheritance
- ☐ Data Abstraction
- ☐ All of the above answers
- ☐ None of the above answers

Q14 - Explain the difference between an Interface, Abstract Class, Class and Object?

Q15 - Optimize the code

If you are a code reviewer and you see the code below, how would you improve/optimize the code below? There are 2 versions to make it easier for you to understand the code (C# and Java).

Java Version

```
public class Car
{
        public void move()
        {
                // move
        }
}

public class Airplane
{
        public void move()
        {
                // move
        }
}

public class VehicleService
{
        public void move(Object a)
        {
                if(a instanceof Car)
                {
                        ((Car)a).move();
                }
                else if(a instanceof Airplane)
                {
                        ((Airplane)a).move();
                }
                else
                {
                        throw new
UnsupportedOperationException();
                }
        }
}
```

C# Version

```
public class Car
{
        public void Move()
        {
                // move
        }
}

public class Airplane
{
        public void Move()
        {
                // move
        }
}

public class VehicleService
{
        public void Move(Object a)
        {
                if(a is Car)
                {
                        ((Car)a).Move();
                }
                else if(a is Airplane)
                {
                        ((Airplane)a).Move();
                }
                else
                {
                        throw new
NotSupportedException();
                }
        }
}
```

Answer: