**1.** What is an application domain?

It is a scope of things that the tool (program/programming language/library) is purposed to do.

**2.** What are ideals for naming?

The name should clearly represent the purpose of the facility. Logically identical operations should have the same name and vice versa.

**3.** What can we name?

Every entity that offers something unique to the user.

**4.** What services does a Shape offer?

It allows to tie objects to the window (to display objects in window), deals with color and style of line and holds the sequence of points and basic notion of how to draw lines.

**5.** How does an abstract class differ from a class that is not abstract?

The abstract class can act only as a base class.

**6.** How can you make a class abstract?

That can be achieved by declaring the constructor as the protected one or declaring at least one pure-virtual function.

**7.** What is controlled by access control?

The access of user and other classes to the members of the class.

**8.** What good can it do to make a data member private?

By making a data member private we can secure it from direct usage and therefore from accidental change of its value. We can provide the function that updates the value of the member but only if the new value doesn’t violate the invariant.

**9.** What is a virtual function and how does it differ from a non-virtual function?

Virtual function is defined within the base class and can be overriden in the derived class. When the value of the derived class is passed to the function that takes the value of the base class as a parameter, overriden in the derived class version of base class virtual function is called.

**10.** What is a base class?

A class used as the base of class hierarchy that typically has one or more virtual functions.

**11.** What makes a class derived?

Derived class can be used in place of the base class and holds all its members.

**12.** What do we mean by object layout?

It is a sequence of object’s data members in which they are stored in memory.

**13.** What can you do to make a class easier to test?

The order of the same function parameters should be identical in all derived classes. That way we can avoid run-time errors.

**14.** What is an inheritance diagram?

A diagram that demonstrates the relationships between base and derived classes.

**15.** What is the difference between a protected member and a private one?

Protected member can be used by derived classes, while private member is accessible only for class own members.

**16.** What members of a class can be accessed from a class derived from it?

Public and protected members.

**17.** How does a pure virtual function differ from other virtual functions?

Pure virtual function must be overriden by the derived class in order to make it non-abstract, while other virtual functions can be left unoverriden. The derived class that doesn’t override it is considered as an abstract one. Although pure virtual functions can be defined within the base class, programmers tend to do it very rare.

**18.** Why would you make a member function virtual?

If derived classes need to use the function that has the same purpose but requires specific implementation of it, the virtual function should be used.

**19.** Why would you make a virtual member function pure?

If the base class doesn’t have any data members and acts mainly as an interface it should have at least one pure virtual member.

**20.** What does overriding mean?

Derived class can have its own implementation of virtual function that was defined within the base class. When the value of the derived class is passed to the function that takes a value of the base class as a parameter, the overriden version of base class virtual function is called.

**21.** How does interface inheritance differ from implementation inheritance?

Interface inheritance means that object of derived class can be used in place of the base one, while implementation inheritance means that derived class can use all the facilties provided by the base class.

**22.** What is object-oriented programming?

A programming paradigm where a class can be built from another one. It is based on the concepts of class inheritance, encapsulation and run-time polymorphism.