Q.) Describe exception handling.

The special processing that may be required when an exception is detected is called exception handling. We define exception to be any unusual event, erroneous or not, which is detectable by either hardware or software and that may require special processing. Exception handling is done by a code unit or segment called an exception handler. An exception is said to be raised or thrown when its associated event occurs. Java uses try-catch keywords to handle exceptions. A try construct includes compound statement called the try clause and a list of exception handlers which are also referred as catch functions. Every catch must have a parameter and the class of the parameter must be a descendant of the predefined class Throwable. The finally clause is placed at the end of the list of handlers just after a complete try construct and it guarantees that some code will be executed regardless of how the execution of a try compound terminates. After a handler has completed its execution, control flows to the first statement following the try construct. A handler may rearise an exception, using a throw without an expression, which lead to exception being propagated.

Q.) What are the characteristics of subprograms?

Three main characteristics of subprograms are the following:

I. Each subprogram has a single entry point.
II. The calling program unit is suspended during the execution of the called subprogram, which implies that there is only one subprogram in execution at any given time.
III. Control always returns to the caller when the subprogram execution terminates

Q.) What is the header of subprograms?

A subprogram header serves several purposes. First, it specifies that the following syntactic unit is a subprogram definition of some particular kind. In languages that have more than one kind of subprogram, the kind of the subprogram is usually specified with a special word. Second, if the subprogram is not anonymous, the header provides a name for the subprogram. Third, it may optionally specify a list of parameters. For example, in Python:
def add (a,b): This is a header of a Python subprogram named add. Here, def implies that following line specifies a subprogram, a and bare parameters of the subprogram.

Q.) Explain pass-by-value parameter passing mode.

When a parameter is passed by value, the value of the actual parameter is used to initialize the corresponding formal parameter, which then acts as a local variable in the subprogram.. Pass-by-value is normally implemented by copy, because accesses often are more efficient with this approach. Pass-by value concept is advantageous because, it is fast for scalars in both linkage cost and access time.

Q.)What are formal and actual parameters?

The parameters in the subprogram header are called formal parameters. They are sometimes thought of as dummy variables because they are not variables in the usual sense. In most cases, they are bound to storage only when the subprogram is called, and that binding is often through some other program variables. Subprograms call statements must include the name of the subprogram and list of parameters to be bound to the formal parameters of the subprogram. These parameters are called actual parameters. They must be distinguished from formal parameters, because the two usually have different restrictions on their forms, and of course, their uses are quite different.

Q.)Define abstract data type.

An abstract data type is a data type that satisfies the following conditions: The representation of objects of the type is hidden from the program units that use the type, so the only direct operations possible on those objects are those provided in the types definition. Declarations of the type and the protocols of the operations on objects of the type, which provide the type's interface, are contained in a single syntactic unit. The type's interface does not depend on the representation of the objects or the implementation of the operation. Also, other program units are allowed to create variables of the defined type.

Q.)Discuss abstract data types in Java.

Java support for abstract data types is similar to those of most other programming languages, with a few important differences. In Java, all objects are allocated from the heap and accessed through reference variables. Methods in Java must be defined completely in a class. A method body must appear with its corresponding method header. Thus, a Java abstract data type is both declared and defined in a single syntactic unit. A Java compiler can inline any method that is not overridden. Definitions are hidden from clients by declaring them to be private.

Q.)What is a java package and what is its purpose?

Java includes a naming encapsulation construct: the package. Packages can contain more than one type definition, and the types in a package are partial friends of one another. Partial here means that the entities defined in a type in a package that either are public or protected or have no access specifier are visible to all other types in the package.

Q.)What is the difference between a class and an instance variable?

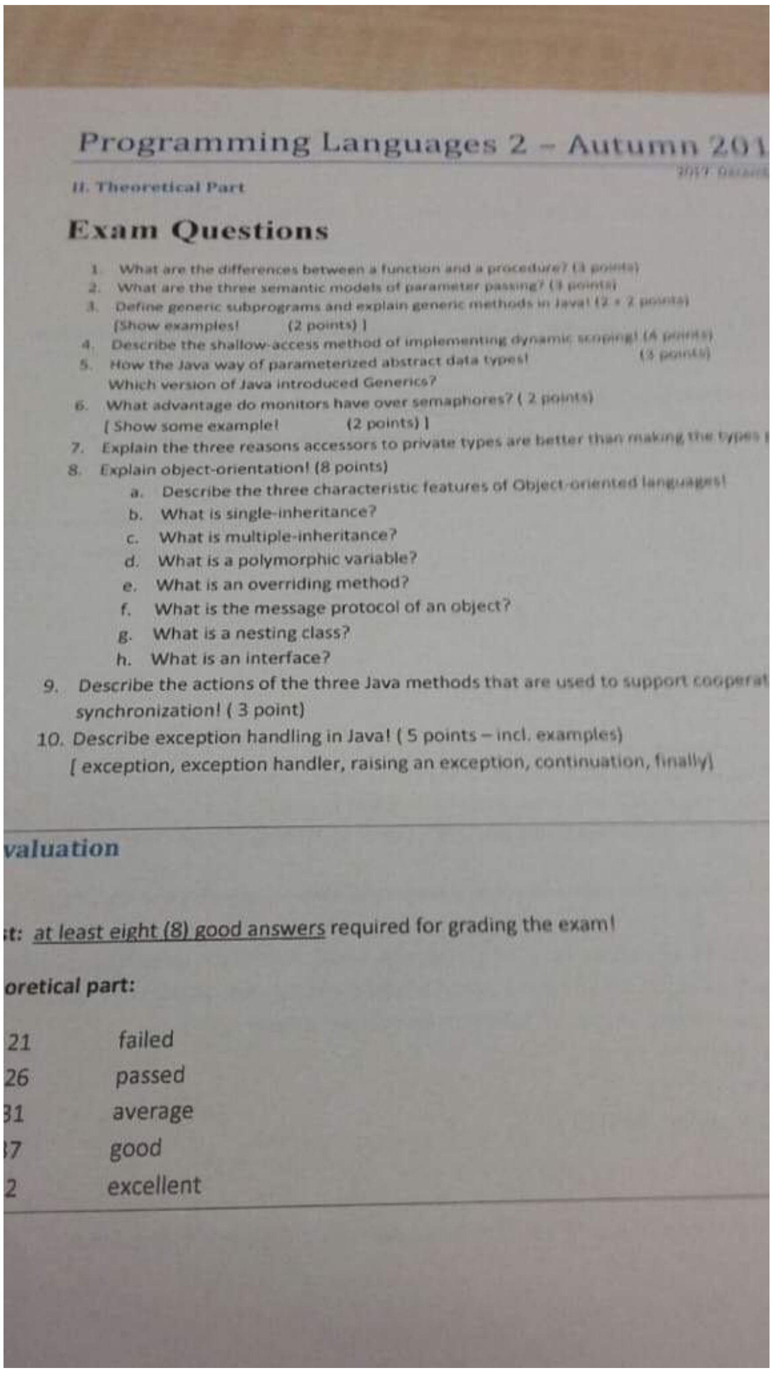
Classes can have two kinds of variables. The most commonly used variables are called instance variables. Every object of a class has its own set of instance variables, which store object's state. The only difference between two objects of the same class is the state of their instance variables. For instance, a class of "person" can have instance variables of name, surname, birthdate. Class variables on the other hand, belong to the class, rather than its object, so there is only one copy for the class. For example, if we wanted to count the number of instances of a class, the counter could not be an instance variable, it would need to be class variable.

Q.) Explain object-orientation.

With object-orientation, programmers can begin with an existing abstract data type and design a modified descendant of it to fit a new problem requirement. The abstract data types in object oriented languages are usually called classes. As with instances of abstract data types, class instances are called objects. A class that is defined through inheritance from another class is a derived class or subclass. A class from which the new class is derived is its parent class or superclass. If a new class is a subclass of a single parent class, then the derivation process is called single inheritance. If a class has more than one parent class, the process is called multiple inheritance. The subprograms that define the operations on objects of a class are called methods. The calls to methods are called messages. The entire collection of methods of an object is called the message interface of the object. Abstract methods are the methods that are declared without an implementation. A class that includes at least one abstract method is called an abstract class. Such a class usually cannot be instantiated. Abstract methods get overridden in the descendant of the class it was declared. A method is said to be overridden, when the subclass modifies the behaviour of its inherited (mentioned) method.

Q.)From where java objects are allocated?

In Java, all objects are allocated from the heap and accessed through reference variables. Most objects are allocated with the new operator, but there is no explicit deallocation operator. Garbage collection is used for storage reclamation.



Programming Languages 2 – Autumn 201

2017.

II. Theoretical Part

Exam Questions

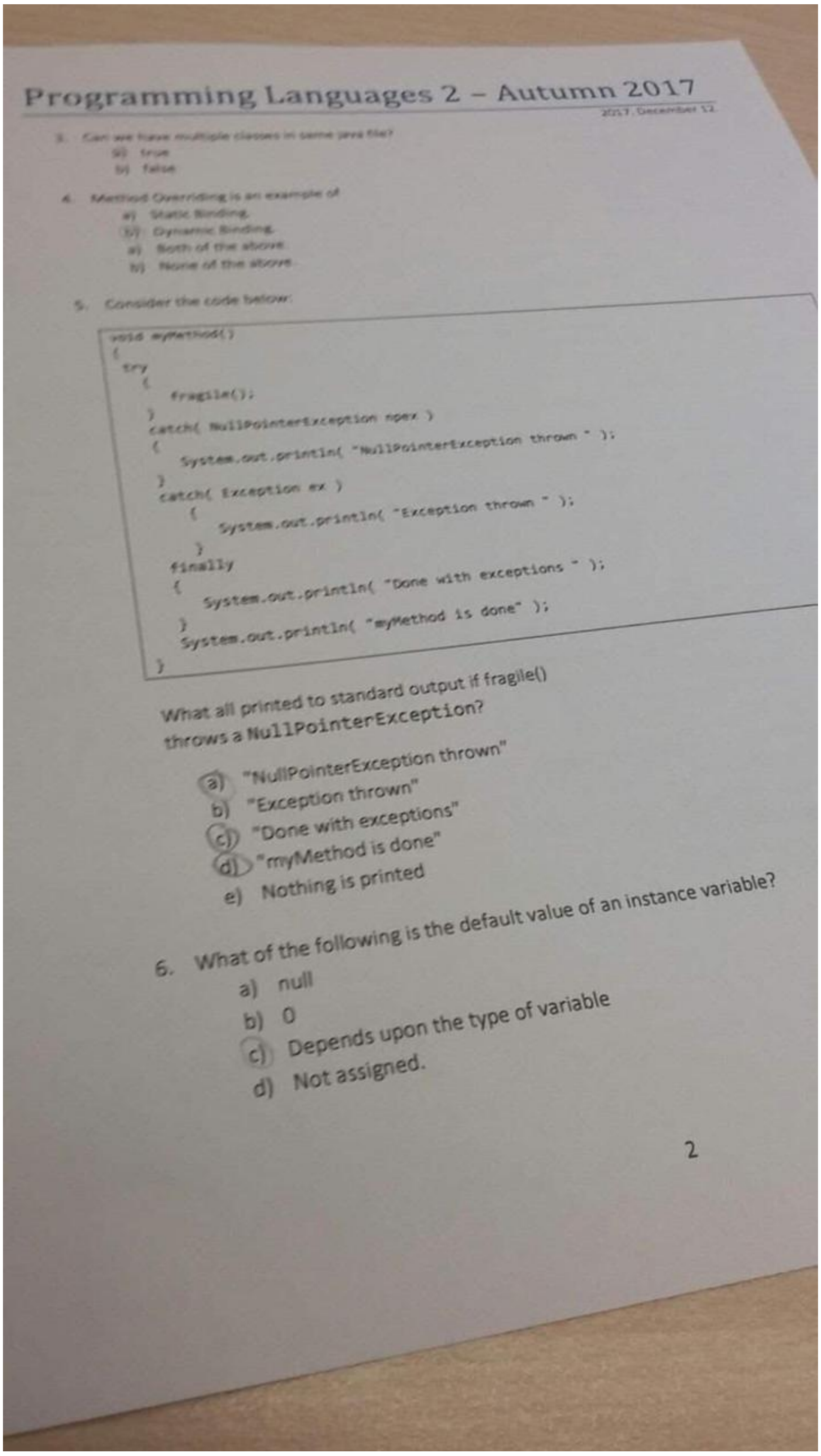
1. What are the differences between a function and a procedure? (3 points)
2. What are the three semantic models of parameter passing? (3 points)
3. Define generic subprograms and explain generic methods in Java! (2 + 2 points)
   [Show examples!   (2 points)]
4. Describe the shallow-access method of implementing dynamic scoping! (4 points)
5. How the Java way of parameterized abstract data types!   (3 points)
   Which version of Java introduced Generics?
6. What advantage do monitors have over semaphores? (2 points)
   [Show some example!   (2 points)]
7. Explain the three reasons accessors to private types are better than making the types
8. Explain object-orientation! (8 points)
   a. Describe the three characteristic features of Object-oriented languages!
   b. What is single-inheritance?
   c. What is multiple-inheritance?
   d. What is a polymorphic variable?
   e. What is an overriding method?
   f. What is the message protocol of an object?
   g. What is a nesting class?
   h. What is an interface?
9. Describe the actions of the three Java methods that are used to support cooperat
   synchronization! (3 point)
10. Describe exception handling in Java! (5 points – incl. examples)
    [exception, exception handler, raising an exception, continuation, finally]

valuation

t: at least eight (8) good answers required for grading the exam!

oretical part:

21	failed
26	passed
31	average
7	good
2	excellent



Programming Languages 2 – Autumn 2017

2017. December 12

3. Can we have multiple classes in same java file?
   a) true
   b) false

4. Method Overriding is an example of
   a) Static Binding.
   [✓ b)] Dynamic Binding.
   a) Both of the above.
   b) None of the above.

5. Consider the code below:

```
void myMethod()
{
  try
    {
      fragile();
    }
    catch( NullPointerException npex )
    {
      System.out.println( "NullPointerException thrown " );
    }
    catch( Exception ex )
      {
        System.out.println( "Exception thrown " );
      }
    finally
    {
      System.out.println( "Done with exceptions " );
    }
    System.out.println( "myMethod is done" );
}
```

What all printed to standard output if fragile() throws a `NullPointerException`?

[✓ a)] "NullPointerException thrown"
b) "Exception thrown"
[✓ c)] "Done with exceptions"
[✓ d)] "myMethod is done"
e) Nothing is printed

6. What of the following is the default value of an instance variable?
   a) null
   b) 0
   [✓ c)] Depends upon the type of variable
   d) Not assigned.

/* Potential Programming Language II (Java) Questions and Answers */

1)a Interpreter does a conversion line by line
as the program run.
2)b super.tesr();
3)a true
4)b dynamic binding
5)a NullPointerException thrown, c Done witj
exceptions, d myMethod is done.
6)c depends upon the type of variable.

Q1.) What are the differences between functions and procedure

1) difference between a function and a procedure, -> A function returns a value, while a procedure does not, it just executes a command

Q2.) What are the 3 semantic models of parameter passing

2) In Mode, Out mode, Inout mode

Q3.) Define the generic subprogram and explain generic methods in Java.

3) Parametric polymorphism is provided by a generic subprogram that take generic parameters that are used in type expressions that describe the types of the parameters of the subprogram. Different instantiations of such subprograms can be given different generic parameters, producing subprograms that take different types of parameters. Parametric definitions of subprograms all behave the same. Support for generic types and methods was added to Java in Java 5.0. Java generic methods differ from generic subprograms of other programming languages in several important ways. First, generic parameters must be classes, that is they cannot be primitive types. Second, although Java generic methods can be instantiated any number of times, only can be instantiated any number of times, only one copy of the code is built. The internal version of a generic method, which is called a raw method, operates on "Object" class objects. At the point where a generic value of a generic method is returned, the compiler inserts a cast to the proper type. Third, in Java, restrictions can be specified on the range of classes that can be passed to the generic method as generic parameters. Such restrictions can be referred as bounds.

Q5.) How the Java way of parametrised abstract data types.

5) Java way of parameterized abstract data types, java uses generics to implement parameterized abstract data, types,

Q5.) Which version of Java introduced Generics

b) Java 1.5 introduced generics

Q6.) What advantage do monitor have over semaphor?

6) Advantage of monitor over semaphore Mutual exclusion in monitor is automatic while mutual exclusion in semaphore needs to be coded explicitly. Shared variables are global to all processes in the monitor while shared variables are hidden in semaphores.

Q7.) Explain the three reasons accessors to private types are better than making types public

7) Encapsulation provides data hiding and more control on the member variables. If an attribute is public then anyone can access it and can assign any value to it. But if your member variable is private and you have provided a setter for it. Then you always have an option to put some constraints check in the setter method to avoid setting an illogical value.

Q8.) Explain object-Orientation

a) Object Oriented languages are languages that encourage reusability and abstraction, three important concepts in object Orientation is inheritance, encapsulation and polymorphism
b) Single inheritance; is when a class extends one class, that is a base class extends only class (super class)
c) Multiple inheritance; is when a class extends more than one class, that is the class has multiple super classes
d) A polymorphic variable; is a variable that has the same interface, but different implementation across classes
e) Overriding a method; is when a child class extends or overrides the implementation of its super class method
f) Message protocol <missing>
g) A nested class; is a class within a class, it can be instance or static class
h) an interface provides definition of method interface without the implementation, which can further be extended

Q10.)Describe exception handling in Java

10) Exception is an error or constraint check that occurs in a program which can stop the program from running, An exception handler handles exceptions if they occur, which prevent programs from failing raising an exception is a process of alerting or throwing an error or raising an exception if an event occurs finally is part of the exception handler that always runs no matter what the result of the exception handler is