CS2510 In-Course Individual Assessment

bought(00245, "Harry potter and the Philosopher's Stone"). bought(00245, "Harry Potter and The Chamber of Secrets"). bought(00245, "hhjbjh"). bought(1, "hjhjh"). customerpurchase(C,Count) :- findall(1,bought(C,\_),L), length(L,Count). bookpurchase(B, Count) :- findall(1,bought(\_,B),L), length(L,Count).

Buy()

Assessment aims at ability to design and implement solutions using programming langs specifically object oriented and logic programming.

Write a report comparing the programming effort – process and the final code.

Evaluation of programming languages used – 1,500 words, 3 pages.

Compare your experience with the two programming langs used:

* Writeability
* Reliability

And the following:

Study

Abtraction

View or representation of an entity that includes only most significant attributes.

Abstract data type – representation of the type is hidden from the program units that use these objects

Declaration of the type and protocols of operations on objects are contained in a single syntactic unit.

Advantages

* Reliability
* Less name conflicts
* Program organisation
* Modifiability
* Separate compilation

Abstraction data type: JAVA

All user-defined types are classes

Objects are allocated from heap and passed through reference variables

Individual entities have access control modifiers rather than clauses

Java has a second coping mechanism, package scope which can be used in place of friends. i.e. All entities in all classes in a package that do not have access control modifiers are visible throughout the package.

Encapsulation constructs

Large programs need means of organisation and partial compilation.

Group subprograms that are logically related into a unit that can be separately compiled. These are encapsulations.

Java packages

Packages can contain more than one class definition, classes in a package are partial friends

Clients of a package can use fully qualified name or use the import declaration

OOP

Inheritance

Productivity increases can come from re-use.

Inheritance allows new classes to be defined in terms of existing ones.

Is-a relationship.

Single inheritance or multiple inheritance

ADT are typically called classes.

Class instances are called objects or instances

A class that inherits is a derived class or a subclass

Class which another class inherits is a base class or superclass

Subprograms that define operations on objects are called methods.

Calls to methods are called messages

Entire collection of methods of an object is called its message protocol or message interface.

Messages have two parts – method name, destination object

Two kinds of variables:

Class variables

Instance variables

Two kinds of methods in a class

* class methods – accept messages to class
* Instance methods – accept messages to objects

Inheritance can be complicated by access controls to encapsulated entities

* Class can hide entities form subclasses
* Class can hide entities from its instances
* Class can hide entities from its instances while allowing its subclasses to see them.

Method overriding: A class can modify an inherited method.

Inheritance vs composition

Composition (has-a)

* A rose has-a petal

Composition allows us to construct new classes using existing ones

Polymorphism

Method overloading

* A class can have two or more methods having the same name, if their argument lists are different

Static – early binding – polymorphic operation is selected at compile time.

Subtyping - inclusion

* Allows a method to be written to take an object of a certain type B, but also work correctly if passed an object that belongs to a type S that is a subtype of B.
* Polymorphic operation is selected at run time
* Relies on upcasting – a form of casting where we cast up the inheritance hierarchy from a subtype to a supertype.

Parametric polymorphism – A method or data type is written in a generic fashion – so values can be handled regardless of their type.

E.g. list with elements of arbitrary type. E.g. generics in Java

Overriding vs. overloading

Overriding-

When you redefine a method that has already been defined in a base class. Resolved at compile time(static)

Overloading-

When you define two methods with the same name distinguished by their method signatures- resolved at compile time(static)

Disadvantages of multiple inheritance

* Languages and implementation complexity
* Potential inefficiency

Advantages

* Convenient and valuable

JAVA

It is an interpreted language

* Compiled into bytecodes
* Yes, this is slower than compiled languages
* Java trades speed for platform independence & programmer safety.

Everything is an object

Method arguments are always passed by value

ObVDjects are not copied – only their references are

Method overloading – but no operator overloading.

No structs.

Nice solution to name collisions – packages

Inherently multi-threaded

Lots of libraries

Access controls

Public

Default

Class members:

Public

Default

Protected

Private

No desctructors!

Garbage collection – When an object has no more references to it, the memory space it occupies in the heap gets reclaimed.

Class variables and methods

Classes can define data members that are associated with the class instead of the object – static

Static methods are also supported. Invoked with the class name, without needing to create an instance of the class.

Inheritance:

Only single inheritances are supported, more with inheritance.

Method overriding is permitted

Methods can be declared final – cannot be overridden

Super keyword can be used to invoke parent class method/constructor

Abstract classes help reduce code dependencies

* To make a class abstract, declare with keyword abstract
* Cannot be used to create objects
* Contain one or more abstract methods – no implementation, must be overridden.

Multiple inheritance via interfaces –

Interface can include only certain method declarations and named constants

Class implements an interface

Generics enable types(classes and interfaces) to be parameters when defining classes, interfaces and methods.

Using generics programmers can implement generic algorithms that work on collections of different types.

Doesn’t support procedural programming – everything must be in a class.

No parentless classes

Interfaces provide support for multiple inheritance

Polymorphism – ad hoc, subtyping, parametric

Data binding is used as “normal” way to bind method calls to method definitions.

OO programming involves four fundamental concepts:

Abstraction, Encapsulation, Inheritance, Polymorphism

Design issues – exclusivity of objects, subclasses and subtypes, type-checking and polymorphism, single and multiple inheritance, dynamic binding, explicit and implicit de-allocation of objects, nested classes.

Java only supports OOP.

Logic programming

Prolog is a declarative programming language.

Programmer specifies goal to the achieved and system works out how to achieve it.

Facts, rules and queries.

Logic: what p(x)-> q(x)

Control: how if A->B then B

Has(owen, jaguar)

Rich(x) <- has(X, jaguar)

Facts + rules: knowledge base

Computation as resolution, a deduction mechanism

ASCII representation <- becomes :-

, is logical “and”

All clauses end with “.”

Prolog programs can be compiled but mostly they are interpreted.

Prolog is dynamically typed.

Single data type: term, which includes subtypes:

Atoms, numbers, variables, compound terms.

Atoms

- a primitive data item.

- Begins with a lowercase letter

- If it contains spaces, surround in single quotes

Numbers

* Int, float, scientific notation

Variables: Begin with a capital letter or underscore symbol. “\_” the anonymous variable. Once a variable gets a value, there is no way to change it. No destructive assignments.

Variables can be aliased – sharing their values with other variables.

We say a variable is instantiated when it has a value, otherwise it is uninstantiated.

Compound terms

FunctorName(subterm1, subterm2)

E.g.

Has(owen, jaguar)

President(usa, trump)

Subterm is a constant(atom, number)/variable/nested term

Compound term: functor and arguments

Number of arguments: arity

* Atom – a compound term with arity 0

Terms can be matched ie. Compared. Two terms match if they are identical or if their variables can be assigned values so that the terms become identical.

Two terms are matched by: traversing trees, comparing nodes, assigning values to varaibles.

Via built in “=” operator

Facts are terms which end with “.”

Has(owen, jaguar).

Has(owen,car(jarguar)).

Date(9, march,2017).

Loves(ann,bob)

Rules:

Rich(X) :- has(X, jaguar)

goodPresident(X):-

hair(X, real),

twitterUser(X,no).

Program structure

Predicate is a collection of clauses with the same functor and arity.

E.g.

Loves(john, mary)

Loves(mary, bill)

A prolog program is a collection of predicates.

Queries

:- Term, Term

?- rich(X).

Unification – weather(foggy) = weather(foggy)

X = Y

Topic = president(trump)

Prolog uses the resolution algorithm to find a solution

Backtracking – find other matching answers

No built in loops, whiles, gotos, if-then-elses

Arithmetic operators, some IO, graphics, internet

If-then-esles are offered with:

p :- q -> r ; s. % if q then r else s

Prolog uses nonmonotonic logic

Facts and rules can be changed at any time – dynamic

Assert() – add fact or rule

Retract() – remove fact or rule

You can change the program at runtime!!

Lists in prolog are allowed!

Represented internally as trees

Last element is always the empty list.