NAME: Misty Baragar CS466-001

CHAPTER: 9

LAB: Lab 9B Ex 1, 2, & 4

ANIMATED FLASHCARDS

1. Abstract data type

2. Array-based implementation

3. Binary search

4. Binary tree

5. Containers

6. Data structure

7. Directed graph (digraph)

8. Edge (arc)

9. External pointer

10. Generic data type (class)

11. Graph

12. Leaf node

13. Linked implementation

14. Linked list

15. Nodes

16. Sorted list

17. Undirected graph

18. Vertex

BOOK EXERCISES

11. C

12. C

13. A

14. B

15. D

16. A

17. D

18. C

19. A

20. C

21. C

22. A

23. A

24. A

39. A

40. C

41. D

42. A

43. B

44. B

47. The hallmark of an assembly language is that each assembly language

instruction is translated into one machine language instruction.

48. An assembler translates assembly-language instructions into machine

code. A compiler translates high-level language instructions into

machine code. The translation of an assembler is one to one: One

statement in assembly language is translated into one statement in

machine code. The translation of a compiler is one to many: One highlevel language instruction is translated into many machine language

instructions.

49. The output from a compiler is a machine-language program. That

program may be stored for later use or immediately executed, but the

execution is a distinct process from the translation. The output from

an interpreter is a solution to the original problem, not a program that

when executed gives you the solution.

54. Programming languages reflect differing views of reality, which we call

paradigms. We use these views (paradigms) to classify the languages.

57. Programs are expressed as the evaluation of functions.

59. expressions

referential transparency(an expression denotes a value

irrespective of context)

absence of state

implicit operational semantics(term-rewriting, unification/resolution)

Functional - functional relationship between input and

output (one output for each input)

Relational - logical relationships between entities in the

problem domain

(many possible solutions

- programs are more like database queries)

61. A Boolean variable is a place in memory, referenced by an identifier,

that can contain true or false.

62. A Boolean expression is a sequence of identifiers, separated by

compatible operators, that evaluates to true or false.

64.

A N D 0 1

0 0 0

1 0 1

67. A data type is the description of a set of values and the basic set of

operations that can be applied to values of the type.

68. Strong typing means that each variable is assigned a data type and

only values of that type can be stored in the variable.

72. A declaration is an instruction to the compiler that associates an identifier with a variable, an action, or some other entity within the

language that can be given a name. The programmer can then refer to

that entity by name.

77. An object in the design phase is an entity that has meaning within the

context of the problem. An object in the implementation phase is an

instance of a class.

78. A class in the design phase is a description of a group of objects with

similar properties and behaviors. A class in the implementation phase

is a pattern for an object.

79. A field names data or actions within a class. A method is a named

action within a class. Thus a field can contain a method.

80. Objects can be related by containment, inheritance, or collaboration.

An object can contain another object as a field. An object can inherit the data and behavior of another object class. An object can collaborate with an object of its own class or another class.

81. Top-down design breaks the problem into successive levels of tasks;

object-oriented design breaks the problem into successive levels of

data objects.

82.

a. Brainstorming, filtering, scenarios, and responsibility algorithms

b. Brainstorming is a group problem-solving activity that involves the

spontaneous contribution of ideas from all members of the group.

The output from this activity is a list of possible classes.

Filtering is a group activity in which the tentative list of classes is

analyzed to determine if there are duplicates, if some classes share

common attributes and behaviors, and if there are classes that

really do not belong in the solution.

Scenarios are group activities that determine the responsibilities of

the classes. They ask “what if” questions and determine if all

possible situations have been considered.

Responsibility algorithms are the algorithms that implement the

responsibilities. This phase is where the algorithms to carry out the

solution get written

c. Brainstorming: a list of possible classes.

Filtering: CRC cards for the classes that survived this stage.

Scenarios: CRC cards with responsibilities outlined and collaborators indicated.

Responsibility algorithms: Algorithms for each of the responsibilities.

d. No, each state is not independent. The first stage produces a tentative list that is used as input to the second stage. The second stage

produces a list of classes that have survived the filtering stages as

input to the third stage. The third stage produces completed CRC

cards that are input to the fourth stage.

83.