Main Object Orientated Programming Concepts

Encapsulation - all data in that class declared private (self-governing)
- for this class to interact with other classes,
constructors or public methods
interface

Abstraction - "What is relevant?"
- Only shows related information

Abstract Data Type (ADT) - set of values and operations

Inheritance - "is a" relationship

Multiple inheritance (not java) has interfaces called by implements

Book (parent) — Dictionary
Book (parent) — Novel (child) — Mystery
Novel (child) — Romance

(Dictionary and Novel are siblings)

```
class Dictionary extends Book {
    super(get a variable from parent class)
}
```

Overloading - 2+ methods can have same name, each different arguments
Overriding - 2 methods that have same name, one in parent, one in child

Polymorphism - "has a" relationship

Example Class Hierarchy

Animal — Dog — Poodle
Animal — Dog — Golden
Animal — Cat
Animal — Horse

```
Animal pet = new Dog();
```

UML Use Case Diagrams

e.g. Space Invaders

Player — Initiator — Use Cases

Time — Player — Move

etc — Fire

Title
- ID
- Initiator
- Preconditions (what needs to be true)
- Flow of Events
- Postconditions (what becomes true)

(Like ~~variables~~ functions interacting)

Relationships

Start Application <<include>> Display Loading Screen

End Game <<extends>> Lost life

e.g. Use Case: Launch Application
ID: Use Case 1
Initiator: Player
Flow of Events: - Player double-clicks space invaders app icon
- System starts up
- <<include>> Display Loading Screen Use Case
- Use Case Ends

Pre-conditions: Application installed
Post-conditions: ~~Introducti~~
Introduction screen displayed

Class Diagrams

Look for game entities — colour
objects — xyz position
tokens — sprite size

e.g. Space Invaders

Class Name
- Attributes
- Operations

Lives - value
Score - value
spacecraft - object
player bullet - object
etc...

e.g.

To Do List
subject description list array
submit() cancel() delete() clear display()

e.g. Player } Class Name
String - Name } Attributes
High_score - int }
Move() } Operations
Fire() }

Relationships

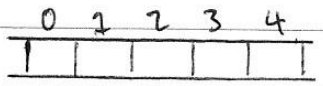
Generalization: connects child to parent { spacecraft → game entity

Association: two classes working together { Bomb — Invader
- can have multiple, so 1)...
2)...
3)...

Dependency: two elements exist for each other { Leaderboard → Player
scores

Structures for Data

Arrays - int[] array = new int[5];



Queue - FIFO

- Initialisation: Length = 0, Front = 0, Back = 0
- Add: Length = 1, Front = 0, Back = 1
- Add: Length = 2, Front = 0, Back = 2
- Remove: Length = 1, Front = 1, Back = 1

Array not full, Length full, Add: Length = 2, Front = 3, Back = 0

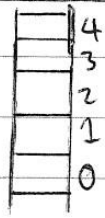
" " , Add: Length = 3, Front = 3, Back = 1

Array full, exit: Length = 5, Front = 3, Back = 3

If back = length, back == 0

If length == 0, exit

Requirements: Make queue empty, test if empty, add to end, remove from front, access front without removing



Stack - LIFO (undo function, going back through history)

- Initialisation: Depth = 0, Front element = ?
- (Add) Push: Depth = 1, Front element = 0
- Push: Depth = 2, Front element = 1
- (Remove) Pop: Depth = 1, Front element = 0

If depth = array length

stack full

Else

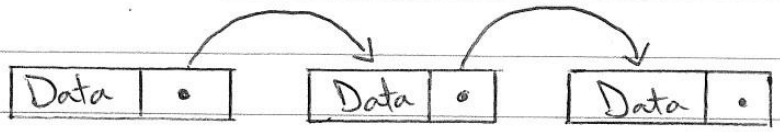
Assign element

Depth ++

Requirements: Make stack empty, test if empty, push to top, pop the top, access top without removing

Linked List - an unbound queue

- links items of data together
- each node contains data and a link to next node



To build a queue data structure using linked lists, 3 classes

- Node
- Queue
- Application

succ = successor

Class Structure

Node

String element;
Node succ;
Node()

Creates new nodes

Queue

int length;
Node front, rear;
addItem (String y)
removeItem()
clear()
isEmpty()
getFirst()
size()

Application

Queue myQueue
main()

Creates a new stack

Clearing, Adding and Removing, think like Queues

Bubble Sort $O(n^2)$

- Looks at neighbouring items and sorts out of place
- Current largest to bottom

1st pass	2nd pass	3rd pass
5, 1, 4, 2, 8	1, 2, 4, 5, 8	Check fully
1, 5, 4, 2, 8	Check	
1, 4, 5, 2, 8	Check	
1, 4, 2, 5, 8	Check	

Selection Sort $O(n^2)$

- Each pass, finds minimum and swaps ~~with~~ if needed

5, 1, 4, 2, 8
1, 5, 4, 2, 8
1, 2, 4, 5, 8
Check

Linear Search $O(n)$

- In sorted array, starts at arr[0] and checks x with elemt
- If found, return index, if not return -1

Binary Search $O(\log n)$

- Takes array size number, half it, check its index.
- Repeat until found x

Exam Preparation Questions 1

1a) Content for mobile devices
- ebooks
- ejournals
- blogs
- elearning content
- native apps
- mobile apps

Software like
- iBooks
- Wordpress

Can insert video, audio, 3D content etc for interactivity

b) Options for distributing content
- Google Play Store
- Apple App Store

Payment channels like
- Paypal
- Google ~~Ring~~ Checkout

c) Take into consideration size of screens, amount of data entry, info, UI/UX

d) Risks of native apps, mobile apps, for Android/Apple - security, payment

2a) Website delivers pages to a browser, while web-service delivers data to program

b) Cross origin policy allows access of web resources on different domains outside of original

c) Native apps approach is based on its platform - Apple iOS default or Android uses individual request, which led to security problems forcing Google to change policy

3a) Testing in emulator, real device, or browser (web app)

c) Android allows GPS (location), Apple uses preconfigured data and menu selections
Both simulate GPS, device cameras, incoming phone calls, SMS, (wireless network communications)

Exam Preparation Questions 2

1a) Cordova executes web-app code, which can also be used in native apps; allows native apps to add extra components. Popular because web developers can build native apps without learning its platform. Only need to use its platform for final build.

b) Each device requires different Cordova API version, so each native app ~~version~~ platform has a web app version that links. Cordova Build service provides an online build service which eleminates above. Different app versions still are needed.

2a) Cross origin policy for browsers doesn't allow JS to access servers in different domains, other than original domain.

b) JSON transmits data in readable format. JSONP returns data as a function call, so bypasses cross origin policy.

c) show([{"name":"Fred", "email":"fred@mail.com", "credit":"5000"}, {etc}])

d) Asynchronous means real-time. AJAX is client-side script provides a 'call-back' for the response.

3a) · Web-server - physical machine or host
· Database server - physical machine or host, Firebase, MySQL, Apache, NodeJS (eg. web server software)
· Server application software - PHP, Python

b) Advantages + Disadvantages of: (scaling opportunities)
Eg Physical server with perma internet connection
Cloud servers (Amazon EC2, Google App Engine, Firebase)