½. Codename1 – framework used to create mobile applications using Java.

* Code is portable and can be built for several platforms, including desktop, Android, and iOS

Objects vs Classes. Object is an instance of a class, class defines the behavior and functionality of the object

Access modifiers – private – class; protected – within same package or derived classes; public – everybody;

Constructors –method same name as class, defines what’s instantiated when object is created.

Vector declaration- **Vector<Integer>=new Vector<Integer>();**

Vector of objects -

private Vector<GameObject> gameObjs = new Vector<GameObject>();

ArrayList ArrayList<Type> str = new ArrayList<Type>();

3. 4 components of OOPs –

Abstraction

* Abstract methods contain method declarations but no method body
  + When you want a class to contain a particular method but want the actual implementation of that method to be determined by a child class
* Abstract classes cannot be instantiated, they are inherited and you implement your functionality for abstract methods in child class.
* You have to extend abstract classes. Circle class extends abstract Shape class.

Polymorphism

* When classes are related to each other by inheritance
* Pig and Dog classes extend Animal class.
* All three classes have animalSound() method implementation.
* Closest (going up to parent class) animalSound() method is called.
  + If dog class doesn’t have animalsound(), then dog object.animalSound() calls the method from animal class instead of dog class as it is normally
* Have object take multiple forms

Inheritance

* Process through which the properties (state and behavior) of parents is reflected in the child. This allows for code reusability and polymorphism

Encapsulation

* A mechanism of wrapping properties together as a single unit. This allows for data hiding and data abstraction.
* Using getters and setters

Information Hiding

* Using setters and getters

Information Bundling

* ????

Association

* An association exists between two classes A and B if instances can send or receive messages (make method calls) between each other

Aggregation

* Represents “has-a” or “is-Part-Of” relationship
* Department has a student, an intramural team is an aggregate of (has) 2 or more students

Composition

* Special type of aggregation
* Has two forms : exclusive and required ownership
* Exclusive – without whole, the part can’t exist
* Required – without part, the whole can’t exist
* Ex. A subscription can’t exist without both a subscriber and a publication (e.g. a

Dependency

* Indicates coupling between classes
* Should try to minimize dependencies
* Other relationships imply dependency

Overloading a method, explain why constructors are commonly overloaded

* Same name but different signatures
* Can occur in same class or split between parent/child classes
* Constructors with different parameters, or different parameter types to construct types of objects given a different set of parameters

Overriding

* when a child class redefines an inherited method with same name, same parameters, and return type
* child objects have code for both methods, but calling overridden method will ALWAYS invoke the child method. Child object can invoke parent method by using super.xxx()

Inheritance

* Process through which the properties (state and behavior) of parents is reflected in the child. This allows for code reusability and polymorphism. Superclass is the parent class, subclass is the child class which inherits methods of the superclass. A class extends another class to inherit that class.

Layouts

GUI

Design Patterns

CN1 Colors

**UML**