Program Development

4 phases

- establishing the requirements:**what**
- creating a design: **how** , which classes and objects are needed
- implementing the code: least creative step
- testing the implementation: Testing attempts to ensure that the program will solve the intended problem under all the constraints specified in the requirements

OBJ desine

The core activity of object-oriented design is determining the classes and objects that will make up the solution

Three of the most common **relationships**:

- Dependency: A uses B

don't want numerous or complex dependencies, or complex classes with out dependency

some happen in same class, like: `str3 = str1.concat(str2);`

(When an object is passed to a method, the actual parameter and the formal parameter become aliases of each other)

- Inheritance: A is-a B

Inheritance allows a software developer to derive a new class from an existing one

- Aggregation: A has-a B

aggregate is an object that is made up of other objects

Class Rules

When a class becomes too complex, it often should be decomposed into multiple smaller classes to distribute the responsibilities

Method Rules

Every method implements an algorithm that determines how the method accomplishes its goals

A potentially large method should be decomposed into several smaller methods as needed for clarity and may use support methods

Program Development (cont)

we shou stop **testing** when we think teh risk of unsoved error is lowenuff

- test case: set of input and user actions, coupled with the expected results
- test suites: formally organized test cases which are stored and reused as needed
- Defect testing: is the execution of test cases
- regression testing: running previous test suites to ensure new errors
- black-box testing: is when a test cases are developed without considering the internal logic
- White-box testing: focuses on the internal structure of the code where that every path through the code is tested

Objects are generally nouns, and the services that an object provides are generally verbs

A good testing effort will include both black-box and white-box tests

Polymorphism

Polymorphism is an object-oriented concept that allows us to create versatile software designs

polymorphic reference: a variable that can refer to different types of objects at different points in time

The method invoked through a polymorphic reference can change from one invocation to the next

All object references in Java are potentially polymorphic

Java allows this **reference to point** to it's constructor object, or to any object of any compatible type

This **compatibility** can be established using inheritance or using interfaces

Assigning a child object to a parent reference is considered to be a widening conversion, and a narrowing for the other way arownd. *(The widening conversion is the most useful)*

An interface name can be used as the type of an object reference variable `Speaker current;` The current reference can be used to point to any object of any class that implements the Speaker interface

Polymorphism (cont)

Selection Sort: find the smallest value in the list switch it with the value in the first position find the next smallest value in the list switch it with the value in the second position repeat until all values are in their proper places

The sorting method needs to be able to call the compareTo method

Insertion Sort:consider the first item to be a sorted sublist (of one item) insert the second item into the sorted sublist, shifting the first item as needed to make room to insert the new addition insert the third item into the sorted sublist (of two items), shifting items as necessary repeat until all values are inserted into their proper positions

Swapping(used for the selection sort algorithm) requires three assignment statements and a temporary storage location:

```
temp = first;
first = second;
second = temp;
```

Recall that an class that implements the Comparable interface defines a compareTo method to determine the relative order of its objects

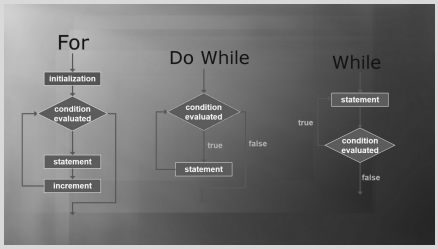
We can use polymorphism to develop a generic sort for any set of Comparable objects

Searching is the process of finding a target element within a group of items called the *search pool*, the taget might not exsist

Linear Search:A linear search begins at one end of a list and examines each element in turn Eventually, either the item is found or the end of the list is encountered

Binary Search:*assumes the list is sorted* then dose the: is hafe value =, < or > then move acordingly, repeet until =

Looping



Arrays

An array is an ordered list of values

An array of size N is indexed from zero to N-1

scores[2] refers the 3rd value in the array

The values held in an array are called **array elements**

An array stores multiple values of the same type the **element type**, The element type can be a primitive type or an object reference

In Java, the array itself is an object that must be instantiated

```
type[] name = new type[n];
```

fixed size = n, index = 0-n-1

If an array index is out of bounds, the Java interpreter throws an *ArrayIndexOutOfBoundsException*

This is called automatic bounds checking

Each array object has a public constant called length that stores the size of the array: name.legth or name.legth() size not index

An **iterator** is an object that implements the Iterator interface

An iterator object provides a means of processing a collection of objects one at a time Several classes in the Java standard class library are iterators (including Arrays)

An iterator is created formally by implementing the Iterator interface, which contains three methods

`hasNext()`: returns a boolean result – true if there are items left to process

`next()`: returns the next object in the iteration

`remove()`: removes the object most recently returned by the next method

float[] prices; = float prices[];

```
int[] units = {147, 323, 89, 933, 540, 269, 97, 114, 298, 476};
char[] letterGrades = {'A', 'B', 'C', 'D', 'F'};
String[] verbs = {"play", "work", "eat", "sleep"};
```

An entire array can be passed as a parameter to a method Like any other object, the reference to the array is passed, making the formal and actual parameters aliases of each other Therefore, changing an array element within the method changes the original An individual array element can be passed to a method as well, in which case the type of the formal parameter is the same as the element type

JavaFX

The GUI designer should:

- Know the user
- Prevent user errors
- Optimize user abilities
- Be consistent

Whenever possible, we should design user interfaces that minimize possible user mistakes

- Error messages should guide the user appropriately
- We should choose the best GUI components for each task
- We should provide multiple ways to accomplish a task whenever reasonable(ctrl-C)

JavaFX programs extend the abstract Application class, inheriting core graphical functionality

launch method sets up the basic graphical interface and calls the start method.(must have this sice inhareted as abstract)
The **main method** is only needed to call the inherited launch method(some IDEs dont need this method)

stage: is the window and holds one scene at a time
scene: set up for what is on the window

the top left corner of the scen is (0,0) all other points are the abs on that like: (112,40)

RGB rang of 0-255

The static rgb method usees specific RGB value

```
Color purple = Color.rgb(183, 44, 150);
```

The color method uses percentages:

```
Color maroon = Color.color(0.6, 0.1, 0.0);
```

GUI in Java is created with at least three kinds of **objects**: *controls*, *events* (is an object that represents some activity to which we may want to respond), and *event handlers* (called a listener)(must accept an ActionEvent object as a parameter)

Basic Shapes

```
Line(startX, startY, endX, endY)
Rectangle(x, y, width, height)
Circle(centerX, centerY, radius)
Ellipse(centerX, centerY, radiusX, radiusY)
Arc(centerX, centerY, radiusX, radiusY, startAngle, arcLength)
```

Translating a shape or group shifts its position along the x or y axis{{ln}} A shape or group can be rotated using the setRotate method

images

```
Image logo = new Image("myPix/smallLogo.png");
imgView.setViewport(new Rectangle2D(200, 80, 70, 60));
```

JavaFX (cont)

A **stack pane** stacks its nodes on top of each other. Since the image view is the only node in the pane, the stack pane simply serves to keep the image centered in the window

A text field allows the user to enter one line of input

All visual components must be attached to a**scene**.
that scene must be attached to a **stage**.
The set of all visual componentsin a scene is called the **scene graph**.
All visual components attached to the scene graph are called **nodes**.
A **branch** node is a node that can contain other nodes.
A **leaf** node is a node which cannot contain other nodes.
The **root** node is the primary container for a scene graph

Branching: if, switch

The order of statement execution is called the **flow of control**. it gose strait though unless:
Repetition statements: execute a statement over and over, repetitively *[if statement, if-else statement, switch statement]*
Conditional statements: decide whether or not to execute a particular statement
[the while loop, the do loop, the for loop]

==	equal to
!=	not equal to
<	less than
>	greater than
<=	less than or equal to
>=	greater than or equal to

||	Logical OR
&&	Logical AND
!	Logical NOT

```
if (sum > MAX)
    delta = sum - MAX;
System.out.println ("The sum is " + sum);
```

If the condition is true, the assignment statement is execif it isn't, it is skipped. *Either way, the call to println is executed next*

nested if statements: if() { if() { } }else { }
An else clause is matched to the last unmatched if (no matter what the indentation implies)

comparing: floats: (Math.abs(f1 - f2) < TOLERANCE)
Characters: a!=A
(0-9 = 48-57, A-Z = 65-90, a-z 97-122)
String: (name1.equals(name2))
(if you use the staight == then: it will only be true if the two String objects are aliases of each other **this is the same for all objects**)

or name1.compareTo(name2) gives a pos neg of 0 out as the compaison value

Branching: if, switch (cont)

```
switch (option)
{
   case 'A':
     aCount++;
     break;
   case 'B':
     bCount++;
     break;
   case 'default':
     cCount++;
     break;
}
```

A switch statement can have an optional default case The default case has no associated value and simply uses the reserved word default If the default case is present, control will transfer to it if no other case value matches If there is no default case, and no other value matches, control falls through to the statement after the switch

The expression of a switch statement must result in an integral type, meaning an int or a char

One **listener object** can be used to listen to two different components
The source of the event can be determined by using the getSource method of the event passed to the listener

Conditional Operator `condition ? expression1 : expression2`

If the condition is true, expression1 is evaluated; if it is false, expression2 is evaluated
The conditional operator is similar to an if-else statement, except that it is an expression that returns a value