ADVANCED JAVA SWEG2033

Advanced Programming

Code: SWEG2033

Chapter One

Revision (Points to Recall)

Prepared By : Befkadu B.(MSc)

ADVANCED JAVA SWEG2033

- Type of Computer Language
- Overview of java Programing
  - History of Java
  - Features of Java
- Object Oriented Programming
  - Object
  - Class
  - Inheritance
  - Polymorphism
  - Abstraction
  - Encapsulation
- Java Modifiers Type
- Java Access Modifiers
- Java Non-Access Modifiers
- Variables
- API, JDK, IDE

ADVANCED JAVA SWEG2033

Types of Computer Language

- Scholars divided Programming languages based on machine and human understanding into three general types:
  1. Machine languages
  2. Assembly languages
  3. High-level languages
- Any computer can directly understand only its own **machine language,** which **is machine dependent**
- **Assembly language** forms English-like abbreviations to represent elementary operations.
- **High-level languages** allow programmers to write instructions that look almost like everyday English and contain commonly used mathematical notations.
- The translators include **assembler, compiler and Interpreter**

3

ADVANCED JAVA SWEG2033

Overview of Java Programming

History of Java

- Java is a high level language.
- 1991 – Sun Microsystems initiates project "Green" with the intent to develop a programming language for digitally controlled consumer devices and computers.
- The language OAK was developed by James Gosling with the goal of using C++'s popularity.
- OAK was later renamed to JAVA and released in 1995.
- Can be used to create two types of programs: applications and applets.

4

ADVANCED JAVA SWEG2033

Overview of Java Programming

Features of Java Programing

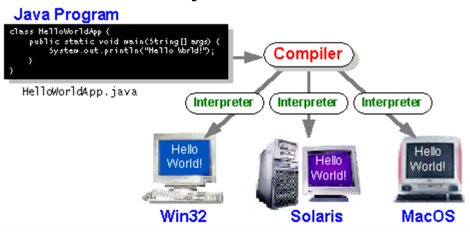
- **Powerful**: massive libraries.
- Java programs are portable.
- Java promise: “Write once, run everywhere”.
- Java differs from other programming languages in that it is both compiled and interpreted language.
- Java compilers produce the Java bytecode.
- Java bytecodes are a set of instructions written for a hypothetical computer, known as the Java virtual machine.
- Bytecodes are platform-independent.
- The JVM is an interpreter for bytecode.

ADVANCED JAVA SWEG2033

Overview of Java Programming

Features of Java Programing (con’t)

- An interpreter reads the byte code and translates into a sequence of commands that can be directly executed by the computer.
- Because the execution of every Java program is under the control of the JVM, the JVM can contain the program and prevent it from generating side effects outside of the system.



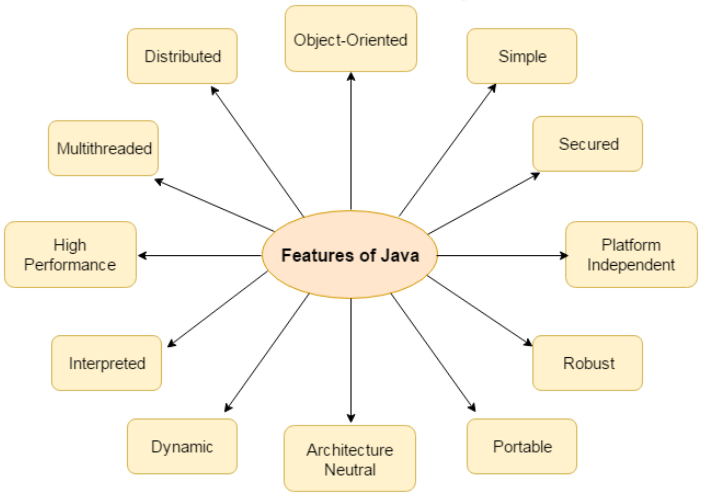
```
class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

HelloWorldApp.java

ADVANCED JAVA SWEG2033

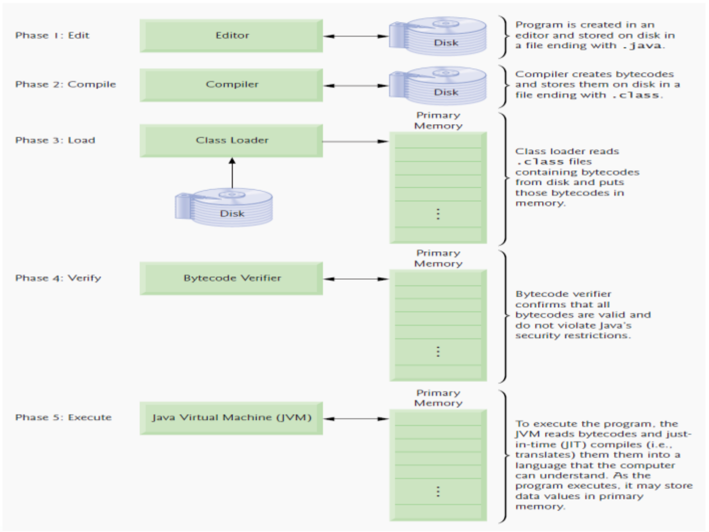
Overview of Java Programming

Features of Java Programing (con't)



ADVANCED JAVA SWEG2033

Overview of Java Programming



Program is created in an editor and stored on disk in a file ending with `.java`.

Compiler creates bytecodes and stores them on disk in a file ending with `.class`.

Class loader reads `.class` files containing bytecodes from disk and puts those bytecodes in memory.

Bytecode verifier confirms that all bytecodes are valid and do not violate Java's security restrictions.

To execute the program, the JVM reads bytecodes and just-in-time (JIT) compiles (i.e., translates) them them into a language that the computer can understand. As the program executes, it may store data values in primary memory.

ADVANCED JAVA SWEG2033

Object Oriented Programming

- In the older styles of programming, a programmer who is faced with some problem must identify a **computing task that needs to be performed** in order to solve the problem.
- Programming then consists of finding **a sequence of instructions that will accomplish that task**.
- But at the heart of object-oriented programming**, instead of tasks we find objects**-entities that have behaviors, that **hold information, and that can interact with one another**.
- Programming consists of **designing a set of objects that model the problem at hand.**

9

ADVANCED JAVA SWEG2033

Object Oriented Programming

- **Object** means a real word entity such as pen, chair, table etc.
- **Object-Oriented Programming** is a methodology or paradigm to design a program using classes and objects. It simplifies the software development and maintenance by providing some concepts:
  - Object
  - Class
  - Inheritance
  - Polymorphism
  - Abstraction
  - Encapsulation

10

ADVANCED JAVA SWEG2033

Object Oriented Programming

Class Activity Q1. What is difference between object-oriented programming language and object-based programming language? With example . You can Discuss next to you.

Have 2 point.

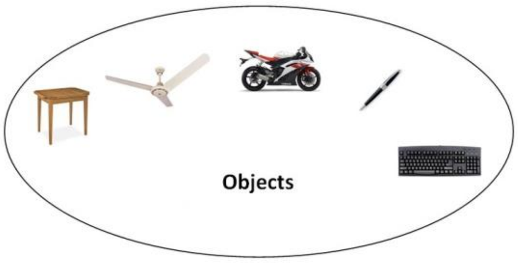
Object based programming language follows all the features of OOPs except Inheritance. JavaScript and VBScript are examples of object based programming languages.

ADVANCED JAVA SWEG2033

Object Oriented Programming

Object

- **Object :** Any entity that has state and behavior is known as an object. For example: chair, pen, table, keyboard, bike etc. It can be physical and logical.



ADVANCED JAVA SWEG2033

Object Oriented Programming

Object

- An object has three characteristics:
  - **state:** represents data (value) of an object.
  - **behavior:** represents the behavior (functionality) of an object such as deposit, withdraw etc.
  - **identity:** Object identity is typically implemented via a unique ID. The value of the ID is not visible to the external user. But, it is used internally by the JVM to identify each object uniquely.
- For Example: Pen is an object. Its name is lexine, color is white etc. known as its state. It is used to write, so writing is its behavior.
- **Object is an instance of a class.** Class is a template or blueprint from which objects are created. So object is the instance(result) of a class.

13

ADVANCED JAVA SWEG2033

Object Oriented Programming

Object

- Represent 'things/objects' from the real world, or from some problem domain (example: "the red car down there in the car park")
- A set of data items (fields) with operations (methods - that implement class behaviors) to manipulate them
- *object* - usually a person, place or thing (a noun)
- *method* - an action performed by an object (a verb)
- Has characteristic behavior.
- Combines data and operations in one place

14

ADVANCED JAVA SWEG2033

Object Oriented Programming

Object

From the above facts Object can be Definitions :

- Object is *a real world entity*.
- Object is *a run time entity*.
- Object is *an entity which has state and behavior*.
- Object is *an instance of a class*.

ADVANCED JAVA SWEG2033

Object Oriented Programming

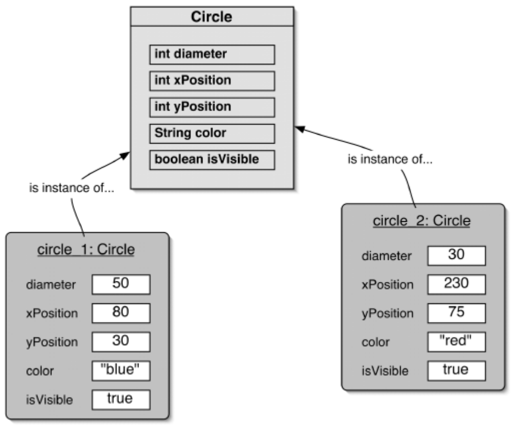
Class

- A class is a group of objects which have common properties. It is a template or blueprint from which objects are created. It is a logical entity. It can't be physical.
- Defines a type of object - a category of similar objects (example: “*automobiles*”)
- Specifies methods and data that type of object has
- Map of a building(class) vs. building (object)
- A class in Java can contain:
  - **fields**
  - **methods**
  - **constructors**
  - **blocks**
  - **nested class and interface**

ADVANCED JAVA SWEG2033

Object Oriented Programming

Class



ADVANCED JAVA SWEG2033

Object Oriented Programming

Class

- **Class Declaration**
  - The syntax for a simple class declaration is

  modifiers class class-name

  {

  body

  }

  - Where the optional modifiers may be one or more of the three keywords {public, abstract, final},
  - Class-name is any valid identifier, and
  - Body is a sequence of declarations of variables, constructors, and methods

ADVANCED JAVA SWEG2033

Object Oriented Programming

Class

Class Activity Q2: 1.5 point

```
class test1 {

   /**
    * @param args the command line arguments
    */
   public static void main(String[] args) {
      // TODO code application logic here
   }

}
```

Is Empty .java file name a valid source file name for the above code?

ADVANCED JAVA SWEG2033

Object Oriented Programming

Class

1. Object and Class Example: main within class

```
class Student{
   int id;
   String name;
     public static void main(String args[])
    {
           Student s1=new Student();//creating an object of Student
           System.out.println(s1.id);
           System.out.println(s1.name);
     }
 }
```

ADVANCED JAVA SWEG2033

Object Oriented Programming

Class

2. Object and Class Example: main outside class

```
class Student {
   int id;
   String name;
}

class TestStudent1{
   public static void main(String args[]){
   Student s1=new Student();
   System.out.println(s1.id);
   System.out.println(s1.name);
 }
}
```

21

ADVANCED JAVA SWEG2033

Object Oriented Programming

Class Activity Q 3 1.5 points

- Ways to initialize object
  1. By reference variable
  2. By method
  3. By constructor
- Initializing object simply means storing data into object.

22

ADVANCED JAVA SWEG2033

Object Oriented Programming

Example for Initialize object by Reference Variable

```
class Student {
   int id ;
   String name;
}
class TestStudent {
   public static void main(String[] args) {
      Student s1 = new Student();
      s1.id = 1234;
      s1.name= "Kebede Megersa Zebre";
      System.out.println("Full Name :" + s1.name );
      System.out.println("ID :       "+ s1.id);
   }

}
```

ADVANCED JAVA SWEG2033

Object Oriented Programming

Example Initialize object by method

In this example, we are creating the two objects of Student class and initializing the value to these objects by invoking the insertRecord method. Here, we are displaying the state (data) of the objects by invoking the displayInformation() method.

```
class Student{
 int rollno;
 String name;
 void insertRecord(int r, String n){
  rollno=r;
  name=n;
 }
 void displayInformation(){System.out.println(rollno+" "+name);}
}
class TestStudent4{
 public static void main(String args[]){
  Student s1=new Student();
  Student s2=new Student();
  s1.insertRecord(111,"Kasa Haylu");
  s2.insertRecord(222,"Lechiso Gebremeskel");
  s1.displayInformation();
  s2.displayInformation();
 }
}
```

ADVANCED JAVA SWEG2033

Object Oriented Programming

Initialize object by Constructor

First Reading Assignment

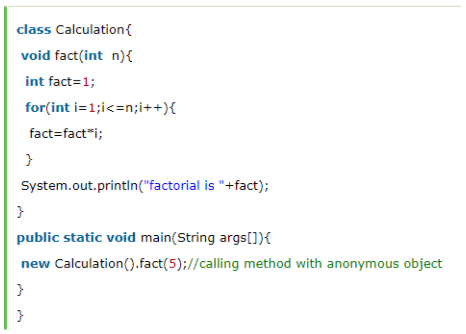
?

ADVANCED JAVA SWEG2033

Object Oriented Programming

Class Activity 4 Q 1.5 points

- Anonymous object
  - Anonymous simply means nameless. An object which has no reference is known as anonymous object. It can be used at the time of object creation only.



```
class Calculation{
 void fact(int  n){
  int fact=1;
  for(int i=1;i<=n;i++){
   fact=fact*i;
  }
 System.out.println("factorial is "+fact);
}
public static void main(String args[]){
 new Calculation().fact(5);//calling method with anonymous object
}
}
```

ADVANCED JAVA SWEG2033

Object Oriented Programming

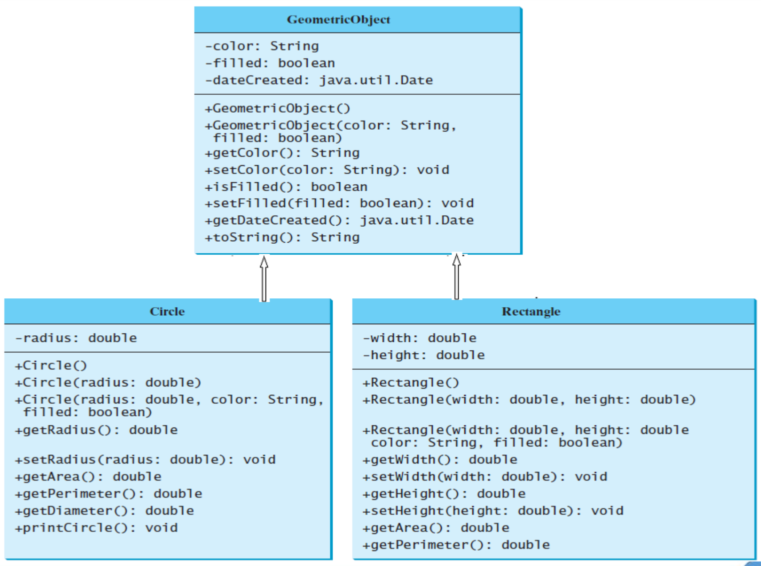
Inheritance

When one object acquires all the properties and behaviors of parent object i.e. known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.

- Inheritance is the process by which one object acquires the properties of another object.
- Allows reuse of implementation
- Classical Inheritance: “is-a” relationship
- Example: fruits and types of fruit (an apple is a type of fruit)
- Goal of inheritance: code reuse

ADVANCED JAVA SWEG2033

Object Oriented Programming


GeometricObject
-color: String
-filled: boolean
-dateCreated: java.util.Date
+GeometricObject()
+GeometricObject(color: String, filled: boolean)
+getColor(): String
+setColor(color: String): void
+isFilled(): boolean
+setFilled(filled: boolean): void
+getDateCreated(): java.util.Date
+toString(): String
Circle
-radius: double
+Circle()
+Circle(radius: double)
+Circle(radius: double, color: String, filled: boolean)
+getRadius(): double
+setRadius(radius: double): void
+getArea(): double
+getPerimeter(): double
+getDiameter(): double
+printCircle(): void
Rectangle
-width: double
-height: double
+Rectangle()
+Rectangle(width: double, height: double)
+Rectangle(width: double, height: double color: String, filled: boolean)
+getWidth(): double
+setWidth(width: double): void
+getHeight(): double
+setHeight(height: double): void
+getArea(): double
+getPerimeter(): double

ADVANCED JAVA SWEG2033

Object Oriented Programming

Polymorphism

- When **one task is performed by different ways** i.e. known as polymorphism. For example: to convince the customer differently, to draw something e.g. shape or rectangle etc. In java, we use method overloading and method overriding to achieve polymorphism.
- The ability to take more than one form
- In terms of the OOP, this means that a particular operation may behave differently for different sub-classes of the same class.
- Informally: same instruction means different things to different agents
  - E.g. “write a note” is similar but different on:
    - Computer
    - Paper
- Technically: many objects can implement same interface in their own way (writing to screen vs. file)

ADVANCED JAVA SWEG2033

Object Oriented Programming

Abstraction

- Hiding internal details and showing functionality is known as abstraction. For example: phone call, we don't know the internal processing. In java, we use abstract class and interface to achieve abstraction.
- Ignore details when appropriate
  - Think about what a method/object/class does, not how it does it
- One of the fundamental ways in which we handle complexity

ADVANCED JAVA SWEG2033

Object Oriented Programming

Encapsulation

Binding (or wrapping) code and data together into a single unit is known as encapsulation. For example: capsule, it is wrapped with different medicines. A java class is the example of encapsulation. Java bean is the fully encapsulated class because all the data members are private here.

- Is the mechanism that binds together code and the data object manipulates
- Put related things in the same place
  - I.e. group related data and operations in an object
  - Each object has its own data and knows how to use it
- Hide internal representation/implementation
- Deny external access to internal fields/methods
- Also called information hiding

31

ADVANCED JAVA SWEG2033

Object Oriented Programming Example

- For example, suppose you want to write a program that automates the video rental process for a local video store.
- The two main objects in this problem are the **video** and the **customer**.
- Then, specify for each object the **relevant data** and possible **operations** to be performed on that data.
- For a video object, the data might include: movie name, starring actors, producer, production company, number of copies in stock
- Some of the operations on a video object might include:
  - checking the name of the movie
  - reducing the number of copies in stock by one after a copy is rented
  - incrementing the number of copies in stock by one after a customer returns a copy

32

Object Oriented Programming

- Object-oriented programming methods aim to achieve the following:
  - To simplify the design and implementation of complex programs.
  - To make it easier for teams of designers and programmers to work on a single software project.
  - To enable a high degree of reusability of designs and of software codes.
  - To decrease the cost of software maintenance.
  - Simplify communication of object
  - Security through data hiding

Object Oriented Programming Summary

- To summarize the OO approach:
  - Emphasis on data, not procedure
  - Programs are made up of objects
  - Data is hidden from external functions
  - Objects can communicate with each other through methods
- The basic concepts of OO are:
  - Objects and classes
  - Data abstraction and encapsulation
  - Inheritance
  - Polymorphism

ADVANCED JAVA SWEG2033

Object Oriented Programming Summary

- In OO, data is critical – data is not allowed to move freely around a system, but it is tied closely to the functions that operate on it.
- Data is protected because the data belonging to an object can only be accessed and modified by the methods of that object.
- Different objects can 'talk' to each other through their methods. In this way, Object A cannot directly modify data belonging to Object B–but it can call a method of Object B that in turn modifies the data.

ADVANCED JAVA SWEG2033

Java Modifiers Type

- Modifiers are keywords that you add to those definitions to change their meanings. Java language has a wide variety of modifiers, including the following:
  - Java Access Modifiers
  - Non Access Modifiers

ADVANCED JAVA SWEG2033

Java Modifiers Type

Java Access Modifiers

Java provides a number of access modifiers to set access levels for classes, variables, methods, and constructors. The four access levels are:

- ➢Visible to the package, the default. No modifiers are needed.
- ➢Visible to the class only (private).
- ➢Visible to the world (public).
- ➢Visible to the package and all subclasses (protected).

37

ADVANCED JAVA SWEG2033

Java Modifiers Type

Java Non-Access Modifiers

Java provides a number of non-access modifiers to achieve many other functionalities.

- ➢ The static modifier for creating class methods and variables.
- ➢ The final modifier for finalizing the implementations of classes, methods, and variables.
- ➢ The abstract modifier for creating abstract classes and methods.
- ➢ The synchronized and volatile modifiers, which are used for threads.

38

ADVANCED JAVA SWEG2033

Java Modifiers Type

Class Activity Q 5

- Is it possible to give private or protected modifiers to java class? 1.5 points
- Why java main method modifier public? 2.5 Points

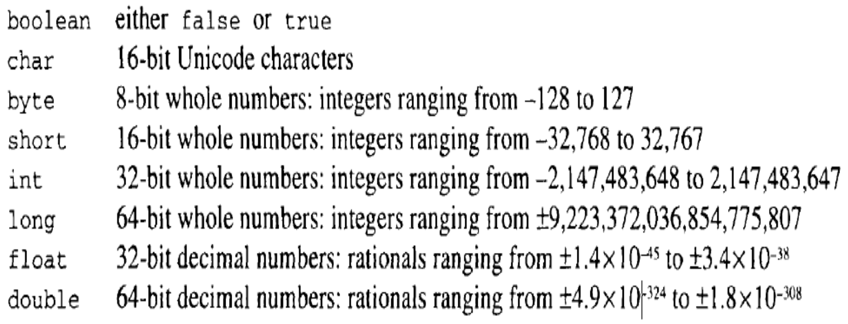
ADVANCED JAVA SWEG2033

Variables

- Variables are locations in memory in which values can be stored. They have a name, a type, and a value.
- A variable has a type and holds a single value while the object may contain many variables
- Every variable has a unique name which is designated when the variable is declared.
- A variable is created when it is declared, and it remains alive until the method in which it is declared terminates.
- The syntax for declaring a variable of any type is:
  - ***type-name variable-name***

Variables (cnt'd)

- Besides reference types, there are eight other types in Java. These are called *primitive types*, to distinguish them from reference types - to store the locations of objects in the computer's memory.
- Their names and values are:



`boolean`	either `false` or `true`
`char`	16-bit Unicode characters
`byte`	8-bit whole numbers: integers ranging from –128 to 127
`short`	16-bit whole numbers: integers ranging from –32,768 to 32,767
`int`	32-bit whole numbers: integers ranging from –2,147,483,648 to 2,147,483,647
`long`	64-bit whole numbers: integers ranging from ±9,223,372,036,854,775,807
`float`	32-bit decimal numbers: rationals ranging from $\pm1.4\times10^{-45}$ to $\pm3.4\times10^{-38}$
`double`	64-bit decimal numbers: rationals ranging from $\pm4.9\times10^{-324}$ to $\pm1.8\times10^{-308}$

Variables (cnt'd)

Class Activity Q6

- What is Instance variable in Java ? 1.5 point
- A variable which is created inside the class but outside the method, is known as instance variable. Instance variable doesn't get memory at compile time. It gets memory at run time when object(instance) is created. That is why, it is known as instance variable.

ADVANCED JAVA SWEG2033

API, JDK, IDE

- Computer languages have strict rules of usage.
- If you do not follow the rules when writing a program, the computer will be unable to understand it.
- The Java language specification and Java API define the Java standard.
- The *Java language specification is a technical definition of the language that includes the* syntax and semantics of the Java programming language.
- The *application program interface (API) contains predefined classes and interfaces for* developing Java programs.
- The Java language specification is stable, but the API is still expanding.

ADVANCED JAVA SWEG2033

API, JDK, IDE

- Java comes in three editions:
  - *Java Standard Edition (Java SE)* - to develop client-side standalone applications or applets.
  - *Java Enterprise Edition (Java EE)* - to develop server-side applications, such as Java servlets and JavaServer Pages.
  - *Java Micro Edition (Java ME)* - to develop applications for mobile devices
- There are many versions of Java SE. The latest is Java SE 8.
- **JDK** consists of a set of separate programs, each invoked from a command line, for developing and testing Java programs.
- Besides JDK, you can use a Java development tool (e.g., Net-Beans, Eclipse, etc) – provides IDE

ADVANCED JAVA SWEG2033

Assignments

- More Read about:
  - Constructors
  - Method Override vs. overload
  - *Abstract class vs. interface*

ADVANCED JAVA SWEG2033

Review Exercise

1. Implement quadratic equation of the form *ax2+**bx +c =0*** by considering a,b,c as random numbers. Handle all special cases
2. Write and run a Java program that generates four random integers, determines their minimum, and prints it.
3. Write and run a Java program that generates a random double, determines which quintile of the unit interval it is in, and reports it. A *quintile is one of the five equal sized pieces of the* whole. The quintiles of the unit interval are 0 to 1/5, 1/5 to *2/5, 2/5 to 3/5, 3/5 to 4/5, and 4/5* to 1.

ADVANCED JAVA SWEG2033

Review Exercise

4. Write and run a Java program that generates a random integer and reports whether it is divisible by 2, by 3, or by 5. Hint: *n is divisible by d if the remainder from dividing n by d is 0.*
5. Write and run a Java program that generates a random year between 1800 and 2000 and then reports whether it is a leap year. **A *leap year is an integer greater than 1584 that is either*** divisible by 400 or is divisible by 4 but not by 100. To generate an integer in the range 1800 to 2000, use int year = Math.round(200*x + 1800); where x is a random float. The round( ) method of the Math class returns the integer nearest the float passed to it. The transformation ***y = 200x + 1800 converts a number in the*** range 0 *.5< x < 1 into a number in the range 1800* ***<= y < 2000.***