Problem 1:

Encapsulation:

public class Student {

private int id;

public Student(int id) {

this.id = id;

}

public int getId() {

return id;

}

}

Polymorphism:

class Person{

void talk(){println("I'm a person");}

}

class man extends person{

void talk(){println("I'm a man");}

}

class woman extends person{

void talk(){println("I'm a woman");}

}

public static void main(){

Person Mike = new Person();

Mike.talk(); //print: "I'm a man";

Person Amy = new Person();

Amy.talk(); //print: "I'm a woman";

}

Inheritance:

class Person{

int age;

String name;

public Person(int age, String name){

this.age = age;

this.name = name;

}}

}

class Student extends person{

String studentID;

public Student(int age, String name, String studentID){

Person(age, name);

this.studentID = studentID;

}

}

public static void main(){

Student mike = Student(19, "mike", 12345);

}

Problem 2:

Java wrapper class wraps primitive data type(int, double, char) into class(Integer, Double, Character)

with its own methods. The most important usage of wrapper class is in Java Collections(list, map, etc...)

because it doesn't support primitive data type.

Problem 3:

HashMap stores only individual data, like HashMap<Integer>,

whereas HashTable stores pairs of data, like HashMap<Integer, Integer>.

Problem 4:

Java String pool is a special heap memory location specific used for storing string.

It's useful because it's more memory efficient because we can just store the reference to the strings

instead of creating multiple strings. Also, it's faster in comparison, because we just need to check

if the references are the same.

Problem 5:

Java Garbage Collections is performed by jvm to automatically delete unused objects to free up memory.

In C++, we need to write destructor to destroy the objects to free up memory. But we don't need to do that

in Java, really convenient...

Problem 6:

Access modifiers are keywords to specify the accessibility of a class, variables or methods. For example, Private keywords specifies variables or methods that's only available in its own class. On the other hand, variables with Public keywords can be access by all classes.

Problem 7:

Final keyword: when used for variables, it can't be modified. when used for methods, it can't be overriden. when used with classes, it can't be inherited.

Problem 8:

Static keyword is used so that, in a class, you don't need to re-declare its own objects to use its local variable or methods.

Problem 9:

Overloading is two methods having the same name but different input parameter or return type. Overriding happens in a inheritance relationship when a children class needs to re-define the same methods from its parent class.

Problem 10:

"this" is usually used in constructor to specify the class's local variable. "super" is used in inheritance when a children class wants to access the content from parent class.

Problem 11:

Uncovered yet, Will update after thursday's lecture...

Problem 12:

A good example of Polymorphism is in inheritance relationship, a children class can override methods from a parent class. The same method is in different execution form in different class. For Specific code example, refer to problem 1.

Problem 13:

Good example of Encapsulation is the public and private variables. We need to use it because it can hide a class's information to others for safety or privacy concerns.

Problem 14:

Interface cannot creat its own object, but other classes can implement the interface to access its methods.

Abstract class cannot creat its own object either, but it can has children class in inheritance relationship, the children class has full access to the abstract class.

Problem 16:

The most common interface of Queue is linkedlist and PriorityQueue. A linkedList implemented Queue can be used as normal FIFO queue. A PQ implemented queue can be used when you want to assign specific order for the queue, i.e, smallest element first.