# Unit 5 Discussion: Inheritance and Polymorphism

1. What was the hardest part of Unit 4 assignments? Please explain. (Note: the purpose of this discussion question is for you to reflect on your assignments last week and share some tips <what failed, what worked, …> with the class)

2. What did you find most confusing or difficult about what you read this week? You may also ask for help on a PA/CA or use of zyBook. (Note: be specific. For example, instead of “xx is hard”, identify the topic and illustrate with an example.)

3. Analyze the following code. Assume each class is in its own .java file.

public class Test extends A {

public static void main(String[] args) {

Test t = new Test();

t.print();

}

} // end class Test

public class A {

private String s;

public A(String s) {

this.s = s;

}

public void print() {

System.out.println(s);

}

} // end class A

A. The program does not compile because Test does not have a default constructor Test().

B. The program has an implicit default constructor Test(), but it cannot be compiled, because its super class does not have a default constructor.

C. The program compiles, but it has a runtime error due to method name print. You can’t have a method named print, as there is an API method System.out.print().

D. Nothing is wrong.

4. Analyze the following code. Assume each class is in its own .java file.

public class Test {

public static void main(String[] args) {

B b = new B();

b.m(5);

}

} // end class Test

public class A {

private int i;

public void m(int i) {

this.i = i;

}

} // end class A

public class B extends A {

public void m(String s) {

}

} // end class B

A. The program can't compile, because m is overridden thus B only has a method m(String).

B. The program can't compile, because the method m(int) is hidden in B thus b.m(5) cannot be invoked.

C. The program compiles and runs. B inherits the method m(int) from A and defines an overloaded method m(String) in B.

D. none of the above

5. Explain the difference between the **this** keyword and the **super** keyword. When should each be used? Illustrate with code segment examples.

6. Given the following segment of code,

1) Write a constructor for the UndergraduateStudent class that accepts a name as a parameter and initializes the UndergraduateStudent’s state with that name, an age value of 18, and a year value of 0.

2) Add a setAge method into UndergraduateStudent class that not only sets the age but also increments the year field’s value by one.

// represent a university student

public class Student {

private String name;

private int age;

public Student(String name, int age) {

this.name = name;

this.age = age;

} // end 2-param constructor

public void setAge(int age) {

this.age = age;

} // end setAge(int)

} // end class Student

// partial implementation of class UndergraduateStudent

public class UndergraduateStudent extends Student {

private int year;

…

} // end class UndergraduateStudent

7. Analyze the following code:

public class Test {

public static void main(String[] args) {

Object a1 = new A();

Object a2 = new Object();

System.out.println(a1);

System.out.println(a2);

}

} // end class Test

class A {

private int x;

public String toString() {

return "A's x is " + x;

}

} // end class A

A. The program cannot be compiled, because System.out.println(a1) is wrong and it should be replaced by System.out.println(a1.toString())

B. When executing System.out.println(a1), the toString() method in the Object class is invoked.

C. When executing System.out.println(a2), the toString() method in the Object class is invoked.

D. all of the above

E. none of the above

8. What is the output of the following code?

public class Test {

public static void main(String[] args) {

String s1 = new String("Java");

String s2 = new String("Java");

System.out.print((s1 == s2) + " " + (s1.equals(s2)));

}

}

A. false false

B. true true

C. false true

D. true false

9. There are two questions in this problem.

1) Given two reference variables t1 and t2, and that t1 == t2 evaluates to true, t1.equals(t2) must be \_\_\_\_\_\_\_\_\_\_\_.

A. false

B. true

2) Given two reference variables t1 and t2, and that t1.equals(t2) evaluates to true, t1 == t2 \_\_\_\_\_\_\_\_\_\_\_.

A. is always false

B. is always true

C. may be true or false

10. Override the equals() method for a Card class.

public class Card {

// only two instance data members

private int num; // number value of a card: 2-14

// use 11-14 for J, Q, K, A

private int suit; // suit value of a card:

// 0-3 for "Club", "Diamond", "Heart", "Spade"

... // additional members not shown

// override equals() method

// ADD CODE

} // end class Card

11. Override the equals() method for a GroupOfCards class. Assume we have already overridden the equals() method for the Card class.

public class GroupOfCards {

// one single instance data member

private ArrayList<Card> cards; // list of cards

... // additional members not shown

// override equals() method

// ADD CODE

} // end class GroupOfCards

12. Read equals() of String class (line 964, <http://hg.openjdk.java.net/jdk8/jdk8/jdk/file/687fd7c7986d/src/share/classes/java/lang/String.java>) and explain the code line by line how it works.

13. What is the output of the driver code?

public class Gambler {

private int money;

public Gambler(int m) { money = m; }

public int currentMoney() { return money; }

public void addMoney(int m) { money += m; }

public void work() { money += 100; }

public void play() { money /= 2; }

public void liveAnotherDay() { work(); play(); }

public String toString() { return money+""; }

}// end of class Gambler

public class CompulsiveGambler extends Gambler {

public CompulsiveGambler(int m) { super(m); }

public void work() { /\* do nothing \*/ }

public void play() {

while (currentMoney() > 1)

super.play();

}// end of play

}// end of class CompulsiveGambler

// in client code

Gambler x = new CompulsiveGambler(100);

x.liveAnotherDay();

System.out.println(x);

14. What is the output of the following code? Pay attention to private String getInfo() in Person.

public class Test {

public static void main(String[] args) {

new Person().printPerson();

new Student().printPerson();

}

} // end class Test

class Student extends Person {

private String getInfo() {

return "Student";

}

} // end class Student

class Person {

private String getInfo() {

return "Person";

}

public void printPerson() {

System.out.println(getInfo());

}

} // end class Person

A. Person Person

B. Person Student

C. Student Student

D. Student Person

15. Analyze the following code (Program 1 and Program 2 are each in their own .java file).

// Program 1:

public class Test1 {

public static void main(String[] args) {

Object a1 = new A();

Object a2 = new A();

System.out.println(a1.equals(a2));

}

} // end class Test1

class A {

int x;

public boolean equals(Object a) {

return this.x == ((A)a).x;

}

} // end class A

// Program 2:

public class Test2 {

public static void main(String[] args) {

Object a1 = new A();

Object a2 = new A();

System.out.println(a1.equals(a2));

}

} // end class Test2

class A {

int x;

public boolean equals(A a) {

return this.x == a.x;

}

} // end class A

A. Program 1 displays true and Program 2 displays true

B. Program 1 displays false and Program 2 displays true

C. Program 1 displays true and Program 2 displays false

D. Program 1 displays false and Program 2 displays false

16. Given source code of the ArrayList class, trace the execution of this constructor call:

new ArrayList<>(10)

Explain line by line which method (please also identify line # in a specific source code file) of which class will be called. Trace into each super class.

<http://hg.openjdk.java.net/jdk8/jdk8/jdk/file/687fd7c7986d/src/share/classes/java/util/ArrayList.java>

Source code files for classes within the java.util package may be found at: <http://hg.openjdk.java.net/jdk8/jdk8/jdk/file/687fd7c7986d/src/share/classes/java/util>

17. Analyze the following design scenario to decide what classes are needed and the relationships between classes (if any). Be sure to distinguish is-a (inheritance) vs. has-a (composition) relationship.

A [simplified] system used by the Registrar’s Office of a university needs to manage the following information:

* Course: course number (such as CS252) and title.
* Course session: term (such as F1A 2021), class day and time (specific time or a special value for online class w/o meeting time), and instructor.
* Course enrollment record: registration date, student, and final grade