Exam 1

Graded

Student

AKSHAJ KAMMARI

Total Points

46.5 / 70 pts

Question 1

Inheritance **14** / 20 pts

1.1 **(no title)** **3** / 3 pts

✔ **+ 3 pts** Correct

+ 1 pt Will not compile

+ 2 pts Can't use a subclass type reference (N) to refer to a superclass instance (M)

+ 0 pts Incorrect

-Anirudha

1.2 **(no title)** **0** / 3 pts

+ 3 pts Correct

+ 1 pt Will compile, output is 2

+ 2 pts Since the static type of p is P, pq in P is referenced.

✔ **+ 0 pts** Incorrect

1.3 **(no title)** **3** / 3 pts

✔ **+ 3 pts** Correct

+ 1 pt Will not compile

+ 2 pts a's static type is A, and class A does not have a field y

+ 0 pts Incorrect

1.4 **(no title)** **1** / 4 pts

+ 4 pts Correct

✔ **+ 1 pt** Will not compile

+ 3 pts The instance field p in Q will hide the inherited static field p. Which means all references to p in Q must be made via an instance. So Q.p will not compile since p is referred to using the class name, instead of using an instance.

+ 0 pts Incorrect

The instance field p in Q will hide the inherited static field p.
Which means all references to p in Q must be made via an instance. So Q.p will not compile since p is referred to using the class name, instead of using an instance.

1.5 **(no title)** **3** / 3 pts

✔ **+ 3 pts** Correct

+ 1 pt Will compile, output is 5

+ 2 pts T's toString is called by dynamic binding, inherited getMe() returns 3, which is added to 2.

+ 0 pts Incorrect

1.6 **(no title)** **4** / 4 pts

✔ **+ 4 pts** Correct

+ 1 pt Will not compile

+ 3 pts Class Y will not compile because its default constructor calls super(), but X does not have a no-arg constructor.

+ 0 pts Incorrect

Question 2

Interfaces/Abstract Classes **19** / 22 pts

2.1 **(no title)** **3** / 3 pts

✔ **+ 3 pts** Will compile

+ 0 pts Incorrect

2.2 **(no title)** **3** / 3 pts

+ 3 pts Correct

✔ **+ 1 pt** Will not compile

✔ **+ 2 pts** In SomeI, imethod must have parameter of type String

+ 0 pts Incorrect

2.3 **(no title)** **1** / 3 pts

+ 3 pts Correct

✔ **+ 1 pt** Will not compile

+ 2 pts nstuff can't be called on intf since the static type of intf
is M, and it can only call M methods (mstuff)

+ 0 pts Incorrect

Incorrect explanation. You are not instantiating an interface here. You are instantiating an object of the C Class.
nstuff can't be called on intf since the static type of intf
is M<C>, and it can only call M methods (mstuff)

2.4 **(no title)** **4** / 4 pts

✔ **+ 4 pts** Correct

+ 1 pt Will not compile

+ 3 pts B implements Comparable but because it extends A, it also needs to implement Comparable, which is a conflict because a class can only implement a generic interface with one type. (PS 2, #1.2)

+ 0 pts Incorrect

2.5 **(no title)** **3** / 3 pts

✔ **+ 3 pts** Correct

+ 1 pt Will not compile

+ 2 pts new Q() won't compile because can't create an instance of an interface.

+ 0 pts Incorrect

2.6 **(no title)** **3** / 3 pts

✔ **+ 3 pts** Correct

+ 1 pt Will not compile

+ 2 pts Y inherits abstract method xx() from X and does not override it with an implementaton, so Y must be also declared to be abstract.

+ 0 pts Incorrect

2.7 **(no title)** **2** / 3 pts

+ 3 pts Correct

✔ **+ 1 pt** Will not compile

+ 2 pts compIt only accepts instance of class that either implements Comparable, or is a subclass of a class that implements Comparable. Cls is neither.

+ 0 pts Incorrect

+ 1 pt Partially correct explanation.
compIt only accepts instance of class that either implements Comparable, or is a subclass of a class that implements Comparable. Cls is neither.

Question 3

equals method **8** / 8 pts

3.1 **(no title)** **4** / 4 pts

✔ **+ 4 pts** Correct

+ 2 pts equals(Object) is called

+ 2 pts Since static type of target o is Object, the only matching equals method is equals(Object)

+ 0 pts Incorrect

3.2 **(no title)** **4** / 4 pts

✔ **+ 4 pts** Correct

+ 2 pts equals(Q) is called

+ 2 pts Since static type of target qobj is Q, equals(Object) and equals(Q) are both available, closest match is equals(Q) since parameter static type is Q.

+ 0 pts Incorrect

Question 4

Polymorphism **3** / 8 pts

4.1 **(no title)** **1** / 6 pts

+ 6 pts Correct

✔ **+ 1 pt** Correct iteration with loop

+ 2 pts instanceof to check object type

+ 1 pt Casting to Student

+ 1 pt Not polymorphic

+ 1 pt Because of the explicit type check of instance before calling method.

+ 0 pts Incorrect

Loop:
1. Loop iteration correct
2. instanceof check not used
3. no cast to student before calling getSchool()
Reasoning:
No reasoning given

p.getSchool() wil not work until you use an object which is a Student. You will have to explicitly check whether each object in people is a Student or not before calling the getSchool method. You will then have to cast the Person p variable you are using into a Student and then access the method as follows:
for (Person person: people) {
if (person instanceof Student) {
Student student = (Student)person;
System.out.println(student.getSchool());
}
}
Code is not polymorphic, because we explicitly check type of instance before calling method.

4.2 **(no title)** **2** / 2 pts

✔ **+ 2 pts** Correct

+ 1 pt No modification is needed

+ 1 pt The condition person instanceof Student would be
true for an instance of Student or any subclass of Student, and since HonorsStudent is a subclass of Student, no change is needed.

+ 0 pts Incorrect

Question 5

Interfaces **2.5** / 12 pts

5.1 **(no title)** **0** / 4 pts

+ 4 pts Correct

`public class  Movie implements Comparable<Movie>`

+ 0.5 pts implements

+ 0.5 pts `Comparable`

+ 0.5 pts `<Movie>`

`public int compareTo(Movie m)`

+ 0.5 pts int

+ 0.5 pts compareTo

+ 0.5 pts (Movie m)

+ 1 pt `return length - m.length`

✔ **+ 0 pts** Incorrect

Answer incorrect.
Correct sample solution
public class Movie implements Comparable<Movie> {
//All the other members of the class
public int compareTo(Movie m) {
return length - m.length;
}
}

5.2 **(no title)** **2.5** / 8 pts

+ 8 pts Correct

`public class MoviesByYear implements Comparator<Movie>`

✔ **+ 0.5 pts** implements

✔ **+ 1 pt** `Comparator`

+ 0.5 pts `<Movie>`

`public int compare(Movie m1, Movie m2)`

+ 0.5 pts `int`

+ 1 pt `compare`

+ 0.5 pts `Movie m1`

+ 0.5 pts `Movie m2`

+ 1 pt return m1.year - m2.year

`movieList.sort(new MoviesyYear())`

✔ **+ 0.5 pts** `movieList`

✔ **+ 0.5 pts** `sort`

+ 1.5 pts `new MoviesByYear()`

+ 0 pts Incorrect

CS 213 Spring '24: Midterm Exam 1

This exam is worth 70 points. At the end of the term for grade determination, your score here will be doubled to be out of 140 points, for 14% of the course grade.

Name: _____ **NetID:** _____

1. **Inheritance (20 pts, 3+3+3+4+3+4)**

   For each of the following, tell if the code will compile. If so, what will be be printed? If not, why will it not compile? *Give a one sentence explanation for EITHER outcome.* (No explanation gets max 1 point.) Note: Syntax errors, if any, are inadvertent and should be ignored. Also, code outside of an explicit class is considered to be in the main method of some arbitrary class in the same package. (Fields that are not private are accessible to all classes in same package.)

   (a)
   ```
   public class M { public int m1; }
   public class N extends M { public int n1; }
   N n = new M(); System.out.println(n.m1);
   ```
   No, the variable is of type N which is the subclass of M, and is assigned as an object of the superclass, M.

   (b)
   ```
   public class P { public static int pq=2; }
   public class Q extends P { public int pq; }
   P p = new Q(); System.out.println(p.pq);
   ```
   No, because p is directed to class Q during runtime, but is initialized as static in class P

   (c)
   ```
   public class A { public int x; }
   public class B extends A { public int y; }
   A a = new B(); System.out.println(a.y);
   ```
   No, because a is of type A, and y is a field of class B, which class A does not have access to

   (d)
   ```
   public class P { static int p=5; }
   public class Q extends P { int p; }
   System.out.println(Q.p);
   ```
   No, because p in class Q will not be able to override the static int p in class P

(e)
```
public class S {
    public int getMe() { return 3; }
    public String toString() { return "3"; }
}
public class T extends S {
    public String toString() { return (getMe() + 2) + ""; }
}
S s = new T(); System.out.println(s);
```
Yes, this will output 5 because the dynamic "new T()" overrides the toString() in class S and does 3(getMe()) + 2

(f)
```
public class X {
    public int x;
    public X(int x) { this.x = x; }
}
public class Y extends X {
    public static void main(String[] args) {
        System.out.println("Y");
    }
}
```
No, there must be an explicit call of the constructor in class Y since the constructor of class X has an arguement.

2. **Interfaces/Abstract Classes (22 pts, 3+3+3+4+3+3+3)**

For each of the following, tell if the code will compile. If so, no explanation is needed. *If not, explain why.* Syntax errors, if any, are inadvertent and should be ignored. Also, code outside of an explicit class is considered to be in the main method of some arbitrary class in the same package. For correct *yes* answer, no partial credit. For correct *no* answer, 1 point for saying no, rest for reasoning.

(a)
```
public interface I<T> { void istuff(T t); }
public class MyI implements I<String> {
    public void istuff(String s) { }
}
```
Yes

(b)
```
public interface I<T> { void imethod(T t); }
public class SomeI implements I<String> {
    public void imethod(SomeI sm) { }
}
```
No, the types need to be the same, SomeI would not work

NetID: _____

(c)
```
public interface M<T> { void mstuff(T t); }
public interface N<T> { void nstuff(T t); }
public class C implements M<C>, N<C> { ... // nstuff, mstuff implemented }
M<C> intf = new C(); intf.nstuff(new C());
```

NO, intf seems to be an instance of an interface, which is not allowed

(d)
```
public class A implements Comparable<A> {
   public int compareTo(A a) {return 0;}
}
public class B extends A implements Comparable<B> {
   public int compareTo(B b) {return 0;}
}
```

NO, because class B extends class A and both of them implement Comparable

(e)
```
public interface Q { void qm(); }
public class QI implements Q {
public void qm() { }
}
Q qobj = new Q();
```

No, because there cannot be an instance of an interface

(f)
```
public abstract class X { public abstract void xx(); }
public class Y extends X { }
```

NO, class Y needs to include the xx() function that has been initialized in class X.

(g)
```
public class Cls {
   ...
   public Cls() { ... }   public int compareTo(Cls cls) { ... }
}
public class Comp {
   public static <T extends Comparable<T>> void compIt(T item) { ... }
}
Cls.compIt(new Cls());
```

NO, comparable<T> is implemented incorrectly

3. **equals method (8 pts, 4+4)**

Suppose there is class called `Q` in which a method `boolean equals(Q qobj)` is implemented. Assume the following objects are created:

```
Object o = new Q(); Q qobj = new Q();
```

Which version of `equals` (`equals(Object)` or `equals(Q)`) is called in each of the following? Give a 1-2 sentence explanation for your answer.

(i) `o.equals(o);`

equals(Object), because its static type is "Object"

(ii) `qobj.equals(qobj);`

equals(Q), because its static type is 'Q'

4. **Polymorphism (8 pts)**

A class `Person` has a subclass `Student`. The `Student` class defines a method `public String getSchool()` that is not in the `Person` class.

(a) Suppose a list `List<Person> people` is created and populated with a mix of `Person` and `Student` objects. Write code to iterate over the list and print the school for every student in the list. Is your code polymorphic? Why or why not?

```
Person p = new Student();
For (Person p : people) {
  if (p.getSchool() != null){
    System.out.println( p.getSchool());
  }
  else {
    System.out.println("not a student");
  }
```

NetID: _____

(b) Suppose there is a subclass of `Student` called `HonorsStudent`, and suppose the `people` list includes `HonorsStudent` instances as well. If you are required to iterate over the list and print the school for every honors student as well as for every non-honors student, would your code of (a) need to be modified? If so, write the modified code. If not, explain why no modification is needed.

No, we may be able to use the getSchool() function from the student class, which will be inherited into the HonorsStudent Class as it is a subclass of Student.

5. **Interfaces (12 pts, 4+8)**

```
public class Movie { String name; int year; int length; ... }
```

(a) Show what code you would need to add to the `Movie` class so you can sort a list of `Movie` instances in *ascending order of length* by calling the `java.util.List` sort method as follows:

```
movieList.sort(null);
```

Assume `List<Movie> movieList` has been set up and populated before the call.

```
public class Movie {
    String name;
    int year;
    int length;
    Public Movie (List<Movie> movieList) {
    movieList.sort (length);
    }
}
```

(b) Suppose in a different place in your application, you want to sort the same `movieList`, but this time in *ascending order of year*, again calling the `List` sort method on `movieList`. Write code using the `java.util.Comparator` interface to make this happen, including a call to the `List` sort method. (Note: your application will make these two different calls to sort at different places, one to sort by length, and the other to sort by year, so once your program starts running, both types of sort should be doable in the same run.)

```
Public Class Movie implements Comparator<a>{

movieList.sort(year);

}
```