Name: ________________________________ Score: /30

This is part one of a two-part examination.

For this part, you have 90 minutes.

Each question is worth one mark; for each question, you will score either:

1 mark	(perfect or almost perfect)
0.5 marks	(enough to pass)

or

0 marks	(not enough to pass)

Answer in handwriting ONLY.

And submit by 4:05pm to the learning hub: Activities/Assignments/ExamPart1.

Do not submit the exam late.

Part 2 will be posted at 4:05pm.

1. What is the Java instruction to create a Scanner object that can read input from the keyboard?

2. Why would you make a method final, and also what does it mean to make a method final?

3. Write a class Test which contains a method called “test” which contains a while loop. The loop asks the user to “Type in a String”. If the user types in a String that starts with the word “toronto”, the loop stops; otherwise, it displays the String that the user typed, and loops again.

4. What is wrong with the following class? Why won’t it compile? Fix the error.

```
class Exam{
        private Test test;

        public Test getTest(){
                return test;
        }

        public abstract void testing(){
                System.out.println(“pass or fail”);
        }
}
```

5. <u>Why would you</u> declare a method to be abstract?

6. <u>What does it mean</u> when a class is declared abstract?

7. What does the “protected” access modifier do? Give a very precise definition.

8. What is wrong with the following class? Why won’t it compile? Fix the error.

```
abstract class A{
        private int a;

        protected int getA(){
                return a;
        }
        private abstract void quizMe();
}
```

9. What is displayed to the console when a C object is created (use the following class definitions)?

```
abstract class A{
   public A(){
      System.out.println("A");
   }
   protected void greetings(){
         System.out.println(“b”);
   }
}
```

```
class B extends A{
   public B(){
      System.out.println("C");
   }
   protected void greetings(){
      System.out.println("a");
   }
}
```

```
class C extends B{
   public C(){
      System.out.println("B");
      greetings();
   }
   public void greetings(){
      System.out.println("c");
   }
}
```

Answer:

10. What is displayed to the console when a G object is created (use the following class definitions)?

```
class E{
   public int toString(){
         return 1;
   }
}
```

```
class F{
   public int toString(){
     return -1;
   }
}
```

```
class G{
   private E e;
   private F f;

   public G(){
      e = new E();
      f = new F();
      System.out.println(e);
      System.out.println(f);
   }
}
```

Answer:

11. Modify the code so that everything compiles, and so that the class Z can call the class Y greetings method. Explain your answer.

```
abstract class X{
   public X(){
      System.out.println("X");
   }
   protected abstract void greetings();
}
```

```
class Y extends X{
   public Y(){
      System.out.println("Y");
   }
   protected void greetings(){
      System.out.println("y");
   }
}
```

```
class Z {
   public Z(){
      System.out.println("Z");
      greetings();
   }
   public void greetings(){
      super.greetings();
      System.out.println("z");
   }
}
```

Answer:

12. What is test-driven development?

13. What is wrong with the following class(es)? Why won't it/they compile? Fix the error.

```
class T{
        public void second(){
        }

        protected void first(){
        }
}
```

```
class U extends T{
       public void first(){
       }

       protected void second(){
       }
}
```

14. class "Object" has an equals method. What does it look like? It's a very simple method. Show the code for the class *Object* equals method

15. What is wrong with the following equals method…why doesn't Java recognize that I want to use this one *instead of* the class "Object" equals method? Fix my mistake so that Java does recognize this is what I want to do; also add in some helpful code that would have helped the compiler catch my error.

```
class String{
        public boolean equals(String s){
                return this.equals(s);
        }
}
```

16. Following is a BankAccount class. I consider two BankAccounts “equal” if they have the same account number and the same balance. Override the equals() method so that it behaves in that way. Include the “helpful” code that you listed in question #15, above.

```
class BankAccount
{
        private int balanceCents;
        private String accountNumber;
        public BankAccount(int balanceCents, String accountNumber){
                this.balanceCents = balanceCents;
                this.accountNumber = accountNumber;
        }









}
```

17. What is a Java interface?

18. Write a Java interface called Payable with the following methods.

    public double getMinPaymentDollars()
    public double getFullBalanceOwingDollars()

19. Rewrite the BankAccount class from question 16, above, so that it uses both your Payable interface and also the Comparable interface. In this age of greed and materialism, our BankAccounts are more philosophical in their estimation of “worth”. Bank Accounts with the smallest balances are considered “greater than” other BankAccounts.

20. Write a class called Test whose constructor creates a BankAccount ArrayList and sorts it. Show <u>all</u> of the code for the Test class.

21. Java has an Exception class. List one of its methods.

22. List the name of one Java subclass of class Exception.

23. Create your own subclass of class Exception, called MyException, which can store a message that is retrievable via the methods you listed in question #21, above.

24. Write a class called Name that has firstName and lastName instance String variables. The constructor takes both of these as parameters. If either one is null or the first name is “Jason” and/or the last name is “Harrison”, throw a MyException exception.

    The constructor signature must declare that it throws MyException Exceptions.

    A JavaDoc comment must do the same.

25. Show the code to try to create a Name object. Use the hierarchy we used in class so at least four blocks of code are used.

26. Which of the following is a subclass of RuntimeException, used for unanticipated failures where recovery is unlikely? Circle your answer.

    Unchecked exceptions

    Checked exceptions

27. What is the name of Java's official unit-testing framework?

28. Following is a unit test. Write the class to make all these tests pass. Rename the class to the more-standard naming convention used in class. Also rename all methods to the more-standard naming convention used in class. (this question is worth 3 marks).

```
class X{
        private A a;
        public X(){
                a = new A(); c();d();
        }
        private void c(){
                a.doSomethingElse(5);
                if(a.getSomethingElse() == 5){
                        System.out.println(“pass”);
                }else{
                        System.out.println(“fail”);
                }
        }
        private void d(){
                try{
                        a.doSomethingElse(-5);
                        System.out.println(“fail”);
                }catch(MyOwnException e){
                        if(e.getMessage().equals(“number must be + not -5”)){
                                System.out.println(“pass”);
                        }else{
                                System.out.println(“fail”);
                        }
                }catch(Exception e){
                        System.out.println(“fail”);
                }
        }
}
```