|  |  |
| --- | --- |
| 1. Refer to the Pet, Cat, and Fish classes below.  public class Pet{      private String name;      private String species;            public Pet(String n, String s){              name = n;              species = s;          }            public String getName(){              return name;          }            public String getSpecies(){              return species;          }          public String toString(){              return getName() + " is a " + getSpecies();          }  }  public class Cat extends Pet{      private String breed;      public Cat(String n, String b){          super(n, "Cat");          breed  = b;      }      public void speak(){          System.out.println("Meow, Meow");      }      public String toString(){          String msg = super.toString() + " of breed " + breed;          return msg;      }  }  public class Fish extends Pet{      private String breed;      public Fish(String n, String b){          super(n, "Fish");          breed  = b;      }      public void speak(){          System.out.println("Blub, Blub");      }      public String toString(){          String msg = super.toString() + " of breed " + breed;          return msg;      }  } | |
| 1. For each of the following (i) Indicate whether the statement is valid (V) or invalid (I) (ii) If the statement is not valid, indicate why.  |  |  |  | | --- | --- | --- | | **Statement** | **V/I** | **If “I”, indicate why.** | | Fish f = new Fish("Dory", "Blue Tang"); | **V** |  | | Cat c = new Fish("Fred", "Siamese"); | **I** | **Fish is not a Cat** | | Fish f = new Pet("Nemo", "Clownfish"); | **I** | **Pet is not a Fish** | | Pet p = new Fish("Dory", "Blue Tang"); | **V** |  | | Object o = new Cat("Fred", "Ragdoll"); | **V** |  | | Object o = new Pet("Ravioli"); | **I** | **Parameter mismatch** | | |
|  | /6 |
| 1. Refer to the code block below to indicate what is printed for each of the following statements. If an error occurs write “ERROR” AND indicate why the error occurs.   Pet pet1 = new Pet("Princess", "Gorilla");          Cat cat1 = new Cat("Roscoe", "Maine Coon");          Fish fish1 = new Fish("Nemo", "Clownfish");          Pet fish2 = new Fish("Dory", "Blue Tang");   1. System.out.println(cat1 instanceof Pet);//returns true of cat1 is an instance of Pet   **True**   1. System.out.println(new Cat() instanceof Pet);   **Error. Cat requires two parameters.**   1. System.out.println(pet1);   **Princess is a Gorilla**   1. System.out.println(cat1);   **Roscoe is a Cat of breed Maine Coon**   1. System.out.println(fish2);   **Dory is a Fish of breed Blue Tang**   1. Pet[] fish = new Pet[3];           fish[0] = fish1;          fish[1] = fish2;          fish[0].speak();  **Error. The speak method is not in the Pet class.** | |
|  | /6 |

|  |  |
| --- | --- |
| 2. Refer to the code below,  class A {  public A() {  System.out.println("Inside A's constructor");  }  }  class B extends A {  public B() {  System.out.println("Inside B's constructor");  }  }  class C extends B {  public C() {  System.out.println("Inside C's constructor");  }  }  public class Inheritance {  public static void main(String[] args) {  /\*\* Statements for questions go here \*\*/  }  } | |
| 1. After executing the statement A b = new C();, what is output by the program?   **Inside A’s constructor**  **Inside B’s constructor**  **Inside C’s constructor** | |
|  | /1 |
| 1. After executing the statement B a = new B();, what is ouptut by the program?   **Inside A’s constructor**  **Inside B’s constructor** | |
|  | /1 |
| 1. What is the output of the following statement? System.out.println((new A()) instanceof A);   **Inside A’s constructor**  **True** | |
|  | /1 |
| 1. What is the output of the following statement? System.out.println((new A() instanceof B);   **Inside A’s constructor**  **False** | |
|  | /1 |
| 1. What is the output of the following statement? System.out.println((new C() instanceof B);   **Inside A’s constructor**  **Inside B’s constructor**  **Inside C’s constructor**  **True** | |
|  | /1 |

|  |  |
| --- | --- |
| 3. The following Pet class is used to represent pets and print information about each pet. Each Pet object has attributes for the pet’s name and species.  public class Pet{      private String name;      private String species;            public Pet(String n, String s){              name = n;              species = s;          }            public String getName(){              return name;          }            public String getSpecies(){              return species;          }          public void printPetInfo() {              System.out.print(getName() + " is a " + getSpecies());          }  }  The following Dog class is a subclass of the Pet class that has one additional attribute: a String variable named breed that is used to represent the breed of the dog. The Dog class also contains a printPetInfo method to print the name and breed of the dog.  public class Dog extends Pet{      private String breed;      public Dog(String n, String b){          super(n, "Dog");          breed  = b;      }      public void printPetInfo() {          /\* To be implemented\*/      }  } | |
| 1. Consider the following code segment.   Dog fluffy = new Dog("Fluffy", "pomeranian");  fluffy.printPetInfo();  The code segment is intended to print the following output.  Fluffy is a Dog of breed Pomeranian  Complete the Dog method printPetInfo below. Your implementation should conform to the example above | |
| **super.printPetInfo();**  **System.out.println(" of breed " + breed);** | |
|  | /2 |
| (b) The PetMaker class contains the main method for the program. Write code that could be used to create the following pets,   * A rabbit named Floppy * A dog (whose breed is pug) named Arty | |
| **Pet pet1 = new Pet("Floppy", "Rabbit");**  **Dog pet2 = new Pet("Arty", "Pug");** | |
|  | /2 |
| The PetOwner class below is used to generate a description about a pet and its owner.  The PetOwner constructor takes a Pet object and a String value (representing the name of the pet’s owner) as parameters.  public class PetOwner {      private Pet thePet;      private String owner;      public PetOwner(Pet p, String o) {          thePet = p;          owner = o;      }      public void printOwnerInfo() {            /\* To be implemented \*/  }  }  Assume that pet1 and pet2 were created as specified above in the PetMaker class. The following table demonstrates the intended behavior of the PetOwner class using objects pet1 and pet2.   |  |  | | --- | --- | | **Code Segment** | **Result Printed** | | PetOwner owner1 = new PetOwner(pet1, "Jerry");  owner1.printOwnerInfo(); | Floppy is a rabbit owned by Jerry | | PetOwner owner2 = new PetOwner(pet2, "Kris");  owner2.printOwnerInfo(); | Arty is a dog of breed pug owned by Kris | | |
| (c) Complete the PetOwner method printOwnerInfo below. Your implementation should conform to the examples. Assume that class Dog works as intended, regardless of what you wrote previously. | |
| **thePet.printPetInfo();**  **System.out.println(" owned by " + owner);** | |
|  | /2 |