Инспекция кода метода возведения в степень

Интерфейс: метод **`int pow(int a, int b)`** в классе **`root.pow.Power`**
Требования:

1. Предусловие тривиально, т.е., метод должен работать для всех целочисленных значений своих параметров
2. В качестве результата метод возвращает результат возведения первого аргумента в степень, равную второму, со следующими уточнениями
    a. При нулевом значении второго аргумента и любом значении первого должен возвращаться результат 1
    b. При отрицательных значениях второго аргумента и любом значении первого должен возвращаться результат 1 (т.е. отрицательный второй аргумент приравнивается к 0).
    c. При переполнении (т.е., если точный результат возведения в степень превосходит по абсолютной величине 2^{31}) возвращается результат возведения в степень по модулю 2^{32}.

Описание реализации.
Метод **`int pow(int a, int b)`** класса **`root.pow.Power`** реализует дихотомический алгоритм быстрого возведения в степень. Перед проведением инспекции нужно ознакомится с описанием алгоритма.
Дихотомический алгоритм быстрого возведения целого числа a в степень b состоит в следующем.
Степень b представим в двоичной записи
$b = (b_k b_{k-1} \dots b_1 b_0)_2 = 2^k b_k + 2^{k-1} b_{k-1} + \dots + 2^1 b_1 + 2^0 b_0$
Тогда $a^b = a^{2^k b_k + 2^{k-1} b_{k-1} + \dots + 2 b_1 + b_0} = a^{((\dots((b_k 2 + b_{k-1})2 + \dots +)2 + b_1)2 + b_0} =$
$((\dots((a^{b_k})^2 a^{b_{k-1}})^2 \dots)^2 a^{b_1})^2 a^{b_0}$
Будем вычислять последовательно a_i при i=0..k и r_i при i=-1..k так, что
$r_{-1} = 1$

$r_0 = a^{b_0}$	$a_0 = a$
$r_1 = (a^{b_1})^2 a^{b_0}$	$a_1 = a^2$
$r_2 = ((a^{b_2})^2 a^{b_1})^2 a^{b_0}$	$a_2 = a^4$
…	
$r_k = ((\dots((a^{b_k})^2 a^{b_{k-1}})^2 \dots)^2 a^{b_1})^2 a^{b_0}$	$a_k = a^{2^k}$

При этом получается, что можно последовательно вычислять $a_{i+1} = a_i^2$ и $r_{i+1} = r_i$ при $b_i = 0$ или $r_{i+1} = r_i a_i$ при $b_i = 1$. В итоге r_k дает нужный результат.

Задание

Требуется соотнести представленный алгоритм с кодом метода **`int pow(int a, int b)`** класса **`root.pow.Power`** и либо убедиться, что код работает так, как предписывается алгоритмом (при некоторой разумной интерпретации используемых в методе переменных), либо выявить расхождения.

Инспекция проходит по следующему сценарию.

1. Все студенты делятся на группы, проводящие инспекцию совместно.
   В каждой такой группе может быть 3-6 человек.
   Люди в группе могут играть следующие роли.
   a. Ведущий — его задача направлять обсуждение в группе, чтобы оно не выходило за рамки целей инспекции, и протоколировать принятые решения (выявленные ошибки и замечания к коду, решения принимаются большинством участников группы, если у кого-то есть особое мнение, отличное от принятого решения и он на нем настаивает, оно должно быть записано также).
   b. Алгоритмист — его задача как можно тщательнее понять описание требований и алгоритма и отвечать на возникающие по их поводу вопросы других участников группы. Если он не справляется, вопрос переадресуется преподавателю.
   c. Интерпретатор — его задача как можно тщательнее понять разбираемый код и отвечать на возникающие по нему вопросы других участников группы. Если он не справляется, вопрос переадресуется преподавателю.
   d. Инспектор (это каждый участник группы, независимо от того, играет ли он еще какую-нибудь роль или нет) — его задача анализировать код, описание алгоритма и требований, формулировать замечания и вопросы к различным свойствам кода или к требованиям, искать ошибки и расхождения кода с требованиями и алгоритмом.
2. Все студенты некоторое время изучают требования, описание алгоритма и код. Возникающие у них вопросы и недоумения стоит записывать, чтобы обсудить на общем собрании группы. Алгоритмисты концентрируются на том, чтобы как можно лучше разобраться в требованиях и алгоритме. Интерпретаторы концентрируются на том, чтобы как можно лучше понять логику работы кода.
3. Затем проводятся собрания групп. Каждым собранием руководит ведущий группы. Он предлагает участникам группы в определенном порядке высказывать их замечания и вопросы. на вопросы и замечания по требованиям и алгоритму отвечает алгоритмист, на вопросы и замечания к коду — интерпретатор. Если группа приходит к общему мнению, что замечание является дефектом, расхождением между требованиями и кодом или ошибкой, оно протоколируется ведущим как результат работы группы.
   Если высказывающий замечание соглашается, что оно уже запротоколировано, записывать второй раз его не надо.
   Помимо предоставления возможности всем высказаться ведущий должен проследить, чтобы все аспекты работы кода получили должное внимание, а именно.
   a. Должны быть проанализированы все инструкции, все условные переходы ведущий должен убедиться, что ни одной инструкции кода и ни один переход не пропущен при разборе.
   b. Должна быть проанализирована работа кода в особых случаях: при нулевых или отрицательных, или выделенных каким-то образом в требованиях, или

алгоритмом, или особыми действиями в коде значениях параметров или внутренних переменных (например, если в коде есть деление на x-1, стоит внимательно работбрать его работу при значении переменной x = 1, достижима ли такая ситуация или нет, что будет, если она достигается). Особыми считаются любые действия, при которых возможны числовые переполнения, потеря точности и исключения — выход за рамки представимых чисел, операции с большими и маленькими числами, при которых маленькое число может оказаться проигнорированным в результате, деление на 0, обращение полю или методу по ссылке, которая может быть равно null, обращение к элементам строки, массива или списка за их границами, работа с отсутствующими файлами, работа с содержимым пустых файлов и пр.

В итоге собираются протоколы работы всех групп, где зафиксированы выявленные ими замечания и ошибки.

Ожидаемое описание ошибок

Описание ошибок, обнаруженных студентами, подается в форме текстового документа (формата Word, HTML, TXT или другого общедоступного) и должно содержать следующую информацию.

1. ФИО обнаружившего ошибку
2. Дата и время обнаружения (не обязательно совсем точные)
3. Краткое описание ошибки в виде одного предложения (например, «При некоторых значениях аргументов метод XXX создает исключение YYY»)
4. Указание на нарушаемое ошибкой требование (или требования, описывающие поведение в той ситуации, в которой ошибка происходит)
5. По возможности точное и полное описание условий возникновения ошибки и ее проявлений, которое удается дать, вместе с описанием требуемого поведения (с учетом возможного недетерминизма и трудности выявления всех условий, например, «Если первый аргумент метода XXX является степенью двойки, а второй — отрицателен, метод XXX создает исключение YYY с сообщением “ZZZ”примерно в 70% случаев, хотя в соответствии с требованиями RRR и SSS исключений в такой ситуации возникать не должно, а должен возвращаться результат TTT»)
6. Код как можно более краткого теста, наиболее явно демонстрирующего ошибку (при недетерминизме следует выбирать возможно не самый краткий тест, при котором вероятность проявления ошибки как можно больше)